VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE A ROZPOZNÁNÍ MATICOVÉHO KÓDU V REÁLNÉM ČASE

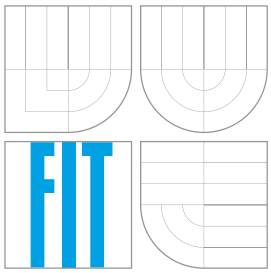
SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE
AUTHOR

Bc. MARTIN DOBROVOLNÝ

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE A ROZPOZNÁNÍ MATICOVÉHO KÓDU V REÁLNÉM ČASE

DETECTION AND RECOGNITION OF MATRIX CODE IN REAL TIME

SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE — Bc. MARTIN DOBROVOLNÝ
AUTHOR

VEDOUCÍ PRÁCE — Doc. Ing. ADAM HEROUT, Ph.D.
SUPERVISOR

BRNO 2012

Abstrakt

Tato práce se zabývá detekcí a rozpoznáním maticových kódů. Experimentuje s využitím PCLines algoritmu. PCLines využívá Houghovu transformaci a paralelní souřadnice pro rychlé hledání přímek v obraze. Navrhovaný algoritmus pomocí dvojitého použití PCLines detekuje sady rovnoběžek a cross-ratio rovnicí řeší problémy obrazu zkresleného paralelní projekcí. Dále popisuje optimalizace pro běh v reálném čase a experimentální implementaci. Výsledky testů ukazují, že využití PCLines je jednou z možných cest k detekci maticových kódů.

Abstract

This work is dealing with detecting and recongnizing matrix codes. It is experimenting use of PCLines algorithm. PCLines is using Hough transform and parallel coordinates for fast detection of lines. Suggested algorithm with double use PCLines detects sets of parallels and problem with image distorted by with parallel projection is solved by cross-ratio equation. We did some optimizations for realtime running and created experimental implementation. Test results shows, that use PCLines one way to detect matrix codes.

Klíčová slova

maticový kód, QR code, detekce, rozpoznání, PCLines, cross-ratio, reálný čas, Houghova transformace

Keywords

matrix code, QR code, detection, recognition, PCLines, cross-ratio, realtime, Hough transform

Citace

Martin Dobrovolný: Detekce a rozpoznání maticového kódu v reálném čase, semestrální projekt, Brno, FIT VUT v Brně, 2012

Detekce a rozpoznání maticového kódu v reálném čase

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Doc. Ing. Adama Herouta, Ph.D.

.....
Martin Dobrovolný
22. května 2012

Poděkování

Děkuji za ochotu, rady a příkladné vedení Doc. Ing. Adamu Heroutovi, Ph.D. Dále také děkuji za vstřícnost a rady Ing. Markétě Dubské.

Obsah

Kapitola 1

Úvod

Detekce a rozpoznání maticového kódu je úlohou v oborech počítačového vidění a zpracování obrazu. Jako velká část úloh v této oblasti nemá jeden ideální algoritmus pro řešení. Existují různá řešení, nejčastěji varianty scanline algoritmu, nebo algoritmy využívající Houghovy transformace.

Tato práce popisuje využití algoritmu PClines vycházejícího z Houghovy transformace. Algoritmus PClines samotný je určen k detekci přímek v obraze. Pomocí vhodných úprav vstupních dat a následně vhodné interpretace stavu Houghova prostoru lze detekovat vzájemně rovnoběžné přímky.

Cílem práce je navrhnout a implementovat algoritmus se snahou o maximální možnou úspěšnost detekce maticových kódů při zachování požadavku na zpracování v reálném čase.

V kapitole 2 je popsán vývoj maticových kódů. Zaměřuje se na jejich nejpoužívanější a zajímavé varianty. Podrobněji se zabývá popisem QR kódu, tento kód je vybrán pro testovací účely navrhovaného algoritmu. Kapitola 3 se zabývá teoretickými znalostmi potřebnými pro navrhovaný algoritmus. Popisuje algoritmus PClines, systém paralelních souřadnic a rovnici cross-ratio. Kapitola 4 se věnuje návrhu vlastního algortimu. Další kapitola 5 popisuje vybrané implementační záležitosti. Následující kapitola 6 se věnuje nastavení různých proměnných algoritmu, dále hodnotí a porovnává dosažené výsledky. Poslední kapitola 7 obsahuje informace o výsledcích práce a naznačuje směry dalšího vývoje.

Kapitola 2

Maticové kódy

Maticové kódy jsou dalším vývojovým stupněm čárových kódů. První čárový kód vznikl dle [7] v roce 1948 jako zakázka na vývoj systému, který bude atomaticky číst informaci o zboží při placení. Společně jej vyvíjeli Bernard Silver a Norman Joseph Woodland. Inspirovali se Morseovým kódem. Čárový kód a různé systémy pro jeho čtení byly úspěšně patentovány v roce 1952.

Čárový kód se pro jeho přednosti velmi rychle rozšířil jak v obchodech, tak v průmyslu. Jednou z jeho předností je jeho relativně malá plocha. S narůstajícími požadavky na množství informací (delší ID výrobku, nebo ukládání dalších informací) uložených v kódu, ale čárový kód postupně rostl. Toto vedlo k evoluci z 1-D kódu na 2-D kód a dle [6] v roce 1988 firma Intermec Corporation vytvořila kód 49. (pozn. Datastrip Code byl vyvinut již v roce 1985, avšak jeho účel byl výrazně odlišný od všech ostatních kódů. Více informací v části 2.3.2).

2.1 Čárový kód

Čárový kód je jednorozměrný, optický, strojově čitelný kód. Právě strojová čitelnost je důvodem jejího užívání. Informace je zakódována v rovnoběžných pruzích a mezerách různé tloušťky. Jako čtecí zařízení slouží speciální optické skenery. V dnešní době umožňují čtení i aplikace mobilních telefonů s fotoaparáty.

Výhody používání čárových kódů:

- rychlost strojového čtení
- jednoduchost obsluhy skeneru
- relativně malá plocha kódu
- praktický přínos - čtecí zařízení okamžitě dostává informaci o konkrétním výrobku, aniž by uživatel musel hledat, nebo číst informace na obalu výrobku

2.1.1 Čtení čárového kódu

Ke čtení čárového kódu slouží zabudované, nebo ruční skenery. Princip jejich fungování je jednoduchý. Jednou z nejpoužívanějších metod je lineární snímání (obr. 2.1). Skener vyšle úzký pruh optického signálu, pomocí kterého získá odezvu kódu připomínající digitální signál. Tento signál je poté snadné dekódovat dle typu čárového kódu.

Obrázek 2.1: Nahoře úzký pruh optického signálu, přečtený signál v dolní části obrázku.

2.2 QR kód

Quick Response code, dále jen QR kód, je nejrozšířenější maticový, tedy 2D kód. Byl vyvinut v roce 1994 firmou DENSO (dnes DENSO WAWE).

Obrázek 2.2: Ukázka QR kódu.

2.2.1 Základní charakteristiky

Charakteristiky QR kódu jsou dány mezinárodním standardem ISO/IES 18004 [1].

Maximální kapacita	
Typ dat	Kapacita
číslice	7089 znaků
alphanumerické znaky	4296 znaků
binární data	2953 bajtů
Kanji	1817 znaků

Tabulka 2.1: Tabulka maximální kapacity QR kódu.

V tabulce 2.1 vidíme maximální kapacitu QR kódu. Kapacita je uváděna při jeho maximální velikosti označované jako 40L (177 x 177 modulů a minimální úroveň opravy chyb). Za alphanumerické znaky jsou považovány číslice, velká písmena A-Z (anglická abeceda), mezera, a znaky podle JIS X 0201.

Velikost kódu se udává v modulech (podrobněji v sekci 2.2.2). Výška i šířka kódu jsou shodné. Podle velikosti rozpoznáváme verze 1 - 40. Verze 1 má rozměry 21 x 21, verze 40 177 x 177 modulů. Velikost kroku mezi verzemi je 4 moduly.

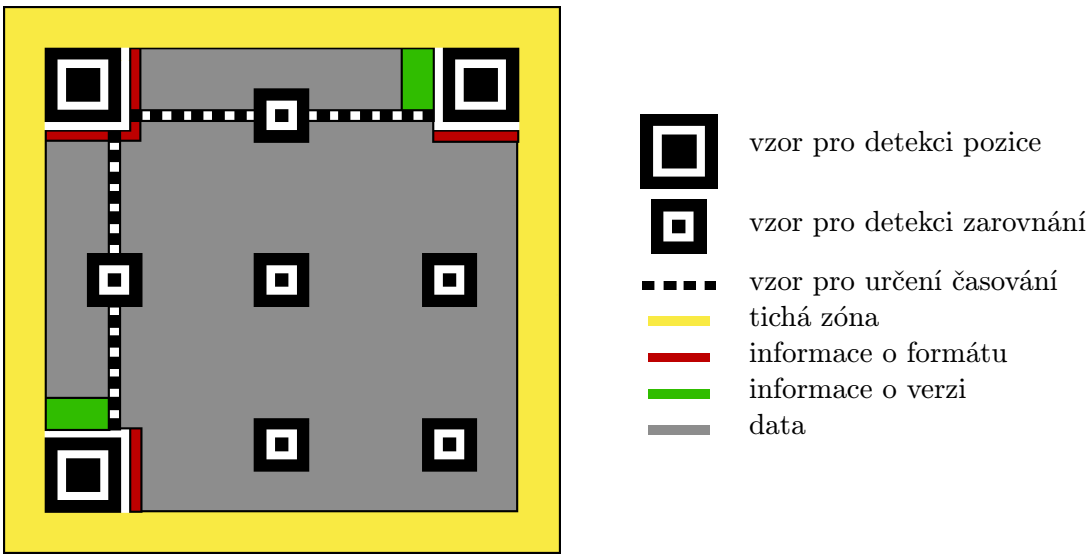
Opravitelnost chyb je udávána jako úroveň opravitelnosti kódových slov, která určuje kolik procent kódových slov může být maximálně poškozeno, aby se data z kódu podařilo rekonstruovat. Přesné hodnoty naleznete v tabulce 2.2.

Tabulka opravitelnosti kódových slov	
Úroveň	Maximální velikost poškození
L	7%
M	15%
Q	25%
H	30%

Tabulka 2.2: Tabulka opravitelnosti kódových slov QR kódu.

2.2.2 Struktura QR kódu

QR kód je tvořen čtvercovými základními jednotkami, které se nazývají moduly. V modulu je zakódován právě jeden bit, může tedy nabývat jedné ze dvou hodnot. Barva modulu v rámci jednoho QR kódu je tmavá, nebo světlá. Obvykle kombinace černá a bílá. Velikost modulu bývá definována délkou hrany a to jako počet obrazových bodů. Velikost modulu se volí s ohledem na předpokládanou kvalitu skeneru. Všechny moduly v QR kódu mají stejnou velikost.



Obrázek 2.3: Schéma rozložení jednotlivých částí QR kódu.

Na obrázku 2.3 vidíme schéma QR kódu. Nejvýraznější jsou rohové značky, vzory pro detekci pozice. Slouží k usnadnění detekce kódu a zjištění jeho orientace. Jejich rozměr je 7 x 7 modulů. Pruhy, kterými je rohová značka spojena s QR kódem, musí být světlé. Vzor pro zarovnání má pomáhat ke korekci zarovnání při čtení kódu. Vzor pro určení časování slouží ke snadnějšímu určení řádků a sloupců. Tichá zóna by neměla obsahovat žádné symboly, ideálně by měla být bílá. Měla by mít tloušťku minimálně 4 moduly. Toto bývá často porušováno. Informace o formátu obsahují informace pro dekódování dat (např. úroveň opravy chyb). Informace o verzi sdělují hlavně velikost QR kódu. Zbytek kódu jsou zakódovaná data a moduly potřebné pro jejích opravu.

2.2.3 Čtení QR kódu

Pro detekci a čtení QR kódů zatím neexistuje optimální, učebnicový algoritmus. Existuje mnoho algoritmů, ty úspěšnější se však stále vyvíjejí. Standard ISO/IES 18004 [1] obsahuje vzorový algoritmus fungující následovně:

1. je vytvořen práh, který je uprostřed mezi nejsvětlejším a nejtmavějším bodem v obraze. Pomocí tohoto prahu je vytvořen dvoubarevný, černobílý obrázek
2. nalezení vzorů pro detekci pozice. Procházíme obrázek po řádcích, hledáme vzor tamvý-světlý-tmavý-světlý-tmavý, jehož části mají relativní poměr 1:1:3:1:1.
3. z poloh rohových značek zjistíme orientaci kódu
4. vypočítáme velikost modulu ((šířka vzoru pro detekci pozice)/7)
5. vypočítáme verzi kódu ze vzdáleností mezi vzory pro detekci pozice
6. z verze kódu určíme jeho velikost strany v modulech a dále i velikost řádku, nebo sloupce
7. extrahujeme bitmapu QR kódu
8. dekódování a oprava dat

2.3 Ostatní maticové kódy

Obsáhlejší výčet maticových, ale i čárových kódů naleznete zde [7].

2.3.1 Kód Aztec

Aztec kód je velmi podobný QR kódu. Je popsán normou ISO/IEC 24778 [3]. Je charakteristický znakem uprostřed obrázku(2.4). Tento znak se nazývá bulls-eye.

(a) kód Aztec (b) jádro kódu Aztec

Obrázek 2.4: Ukázka Aztec kódu a jeho jádra: černá, bílá - jádro, červená - informace o módu kódu.

Bulls-eye má rozměry 9 x 9, nebo 13 x 13. V okolí jeho rohů jsou orientační značky. Tyto značky slouží k určení orientace kódu. Dále je v okolí bulls-eye mezi orientačními značkami informace o módu kódu. Je zde zakódovaná velikost kódu, typ použitého kódování a další informace. Díky informaci o velikosti kód Aztec nevyžaduje tichou zónu tak, jak je tomu u QR kódu.

2.3.2 Kód Datastrip

Datastrip kód, nebo také Cauzin Softstrip, je kód vyvynutý v roce 1985. Umožňuje uložit až 1000 bajtú na palec čtvereční. To je 20x - 100x více než ostatní kódy v té době. Kód byl určen k přenosu programů v tištěné formě, například v časopisech.

Obrázek 2.5: Ukázka Datastrip kódu.

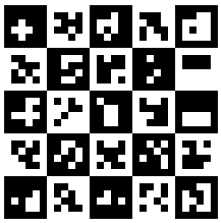
2.3.3 EZCode

EZCode je kódem vyvinutým s ohledem na čtecí zařízení - mobilní telefon s fotoaparátem a přístupem na internet. Byl licencován v roce 2006 firmou Scanbuy. Jeho velikost je vždy 11 x 11 modulů. Tyto moduly se zde nazývají "giant pixels". Umožňuje uložit 83 bitů a jeho velikost je 0,5 palce. Kód sám o sobě nenese žádná užitečná data, pouze ukazatel do databáze společnosti Scanbuy. Výhodou je konstantní relativně malá velikost kódu, avšak pro přístup k informacím je potřeba připojení k internetu.

Obrázek 2.6: Ukázka EZcode.

2.3.4 Grid Matrix

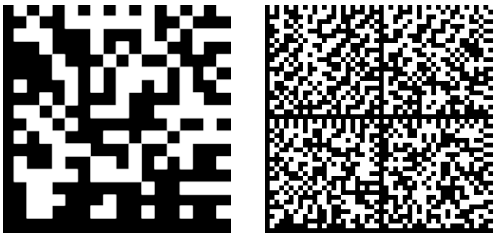
QR kód sice umožňuje jistou míru opravy chyb, avšak poškození rohových značek, zvlněný podklad, nebo poškození informací o kódování, má často fatální následky při jeho čtení. Proto firma Syscantech vyvořila Grid Matrix kód. Značky pro určení pozice jsou v QR kódu umístěny v rozích. To vedlo k vytvoření kódu bez těchto značek. Díky tomu je možné opravit informace obsažené v rohových oblastech. Mohou být poškozeny až všechny 4 rohy. Kód je samozřejmě odolný vůči poškození stran, nebo jiných částí. Kód umožňuje volbu velikosti poškozené plochy, kterou lze opravit. Lze zvolit z pěti úrovní 10%, 20%, 30%, 40% a 50%. Dále umožňuje čtení z libovolého úhlu rotace a až 45° odklon od kolmice na rovinu kódu.



Obrázek 2.7: Ukázka Grid Matrix.

2.3.5 Semacode, Data Matrix

Původní určení semacode je podobné jako u EZCode. Tedy čtecím zařízením by měl být fotoaparát mobilního telefonu. Tato vlastnost zůstává dodnes, ale standard ISO/IEC 16022 [4] se velmi často využívá v průmyslu. Změnou pro průmyslové využití je zmenšení kódu na velikost 2 - 3mm. To samozřejmě vyžaduje i specializované čtecí zařízení.



(a) Semacode (b) Data Matrix

Obrázek 2.8: Ukázka Data Matrix a Semacode.

Na obrázku 2.8 vidíme příklad Semacode čtvercového tvaru. Jeho maximální velikost je 144 x 144 modulů a může obsahovat 2335 znaků. Norma umožňuje i tvar obdélníkový, určuje také jeho konkrétní rozměry. Dále na obrázku vidíme možnost složit více Semacode do jednoho kódu. Tato nově vzniklá struktura se nazývá Data Matrix specifikovaná v normě [12]. Data matrix je určen k přenosu většího množství informací než umožňuje Semacode.

Kapitola 3

PClines

3.1 Houghova transformace

Houghova transformace je matematický nástroj vhodný pro nalezení některých primitiv v obraze. Tato primitiva musí mít známý popis parametrickou rovnicí. Proto se nejčastěji používá pro nalezení přímek, kružnic a nebo elips.

Vstupem do algoritmu jsou hrany detekované ve vstupním obraze. Transformace potom převádí bod z hrany v 2D prostoru na křivku v 3D prostoru. Ve 3D prostoru dále hledáme útvary odpovídající hledaným primitivům. Z jejich polohy v Houghově prostoru jsme schopni určit polohu primitiv ve vstupním obraze.

3.1.1 Hranové detektory

Hranové detektory slouží k identifikaci hran v obraze. Existuje mnoho algoritmů, většina z nich zpracovává jako vstup obraz s jedním kanálem. Je tedy na uživateli, zda převede vstupní obraz na šedotónový, nebo nechá zpracovat každý barevný kanál zvlášť a následně vhodným algoritmem extrahuje hrany využitím výstupů ze všech kanálů.

Hranové detektory dělíme na detektory využívající první derivaci a na detektory využívající druhou derivaci. První derivací se rozumí rozdíl intenzity okolních pixelů. Tyto rozdíly počítáme pomocí konvoluce s konvolučním jádrem specifikovaným dle zvoleného detektoru. Provádíme výpočet derivací pro řádky a sloupce a získáváme tak gradientní obrazy G_x a G_y. Z gradientních obrázů jsme schopni získat informace o velikosti odezvy hrany $G(x, y)$ a její směr θ. Výpočty jednotlivých hodnot jsou znázorněny ve vzorcích 3.1 a 3.2.

$$G(x, y) = \sqrt{G_x^2 + G_y^2} \tag{3.1}$$

$$\theta = arctan\left(\frac{G_y}{Gx}\right) \tag{3.2}$$

Pro získání derivací řádků a sloupců jsou definována různá konvoluční jádra. Hodnoty pro konvoluční jádra jsou definovány pro různé, většinou čtvercové rozměry. Jádra o větších rozměrech bývají více odolná proti šumu v obraze. Hodnoty v jádrech určují chování detektoru. Některá jádra reagují pouze na ostré hrany, jiná reagují i na pozvolnější změnu v obraze. Ukázku různých konvolučních jader naleznete v tabulce 3.1.

Po získání gradientního obrázku jsou jako hrany označeny ty pixely, které mají hodnotu gradientu vyšší, než je hodnota prahu. Velikost prahu bývá určována experimentálně. Lze použít i lokální prahování.

Operátor	Jádro pro G_x	Jádro pro G_y
Rozdíl bodů	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Roberts	$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Prewitt	$\frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
Sobel	$\frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$

Tabulka 3.1: Tabulka kovolučních jader hranových detektorů. Převzato z [11].

Nevýhodou výše zmiňovaných detektorů je, že odezva na neostrou hranu je větší než 1 pixel. Řešením tohoto problému je například Cannyho detektor. Cannyho detektor se snaží o co nejpřesnější lokalizaci hrany. Pro každou hranu existuje pouze jedna odezva. Tzn. odezva na hranu je čára o síle 1 pixel. Dále by měl být dobře odolný vůči šumu, vstupní obraz rozmazává Gaussovským filtrem.

Hranové detektory využívající druhou derivaci detekují hranu při průchodu druhé derivace nulou. Tyto detektory jsou velmi citlivé na šum, trpí ztrátami některých ostrých hran a dalšími problémy. Tyto záporné vlastnosti je dělají nevhodné pro detekci maticových kódů. Proto je zde nebudeme rozebírat.

3.1.2 Houghův prostor

Houghova transformace kromě převodu bodu z 2D obrazu na křivku v 3D prostoru provádí také transformaci z Kartézského souřadného systému na polární. Každý bod ve vstupním obraze může být transformován na křivku v Houghově prostoru.

Je dána normálová rovnice přímky dle 3.3. Pro každou přímku ve vstupním obraze existuje právě jeden bod v Houghově prostoru $[\theta, \rho]$.

Po úpravě rovnice na tvar 3.4 je také zřejmé, že pro každý bod vstupního obrazu na souřadnicích $[x, y]$ po dosazení do rovnice zbývají dvě neznámé. Tyto neznámé θ a ρ vytvářejí v Houghově prostoru již zmiňovaou křivku. Pro kždý úhel θ můžeme vypočítat jeho hodnotu ρ.

$$xcos\theta + ysin\theta - \rho = 0 \tag{3.3}$$

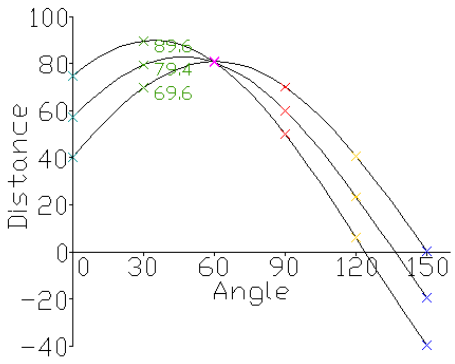
$$\rho = xcos\theta + ysin\theta \tag{3.4}$$

3.1.3 Detekce přímek

Při pohledu na rovnici 3.4 je zřejmé, že v Houghově prostoru mohou vznikat průsečíky mezi různými křivkami. Tyto průsečíky vznikají mezi každými dvěma body, které mohou tvořit přimku v obraze. Vztahy mezi jednotlivými entitami jsou následující (zdroj [9]):

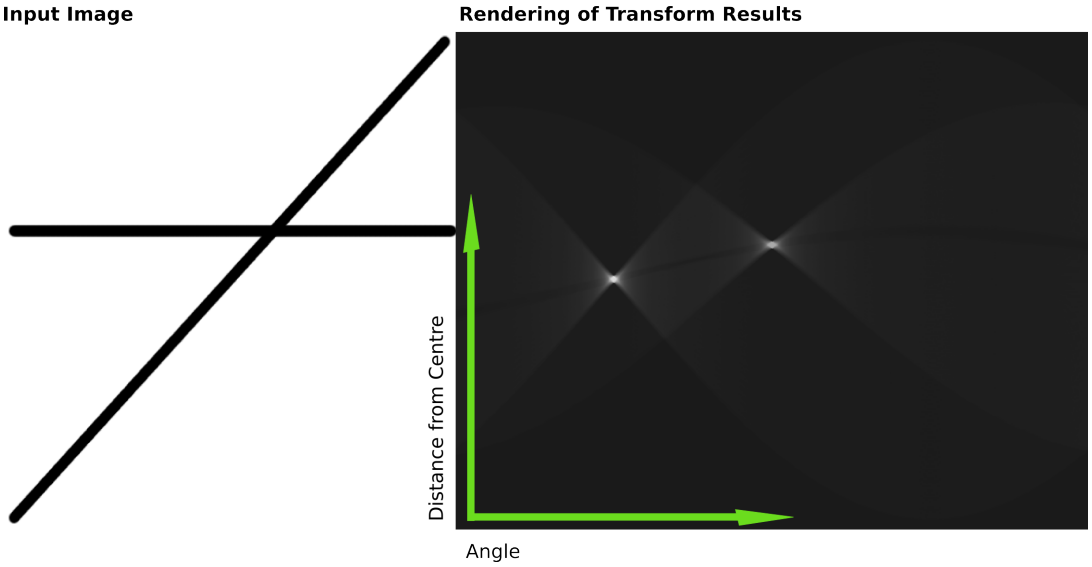
- bod v obráze odpovídá sinusoidě v Houghově postoru
- bod v Houghově prostoru odpovídá přímce ve vstupním obraze
- body ležící na přímce ve vstupním obraze odpovídají křivkám procházejícím společným bodem v Houghově prostoru
- body ležící na stejné křivce v Houghově prostoru odpovídají přímkám procházejícím bodem ve vstupním obraze

Tyto vztahy se také nazývají dualita. Situaci ilustruje obrázek 3.1.



Obrázek 3.1: Ukázka Houghova prostoru po transformaci 3 bodů ležících na přímce. Převzato z [5].

Na obrázku vidíme 3 různé body, ležící na přímce, po Houghově transformaci. Vidíme také vznik průsečíku mezi nimi v místě, kde je θ rovna 60°. Vložení všech hranových bodů do Houghova prostoru popisuje obrázek 3.2.



Obrázek 3.2: Ukázka Houghova prostoru po transformaci dvou přímek. Převzato z [5].

Obrázek 3.2 je ukázkou transformace hranových pixelů dvou přímek. V Houghově prostoru zřetelně vidíme dvě lokální maxima reprezentující tyto přímky. Algoritmickým nalezením těchto maxim lze tyto přímky snadno lokalizovat a zjistit polohu v původním obraze.

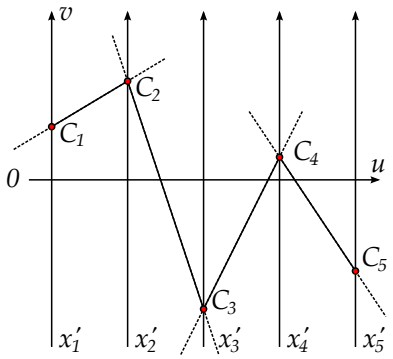
Pro zpětný převod využijeme opět rovnici 3.3. Doplníme-li získané hodnoty θ a ρ, získáme pro každé y v původním obraze právě jedno x. Pozn. výsledné x může ležet mimo obraz, ale stále bude na přímce.

3.2 PClines

PClines je algoritmus určený k detekci přímek v obraze pomocí paralelních souřadnic (zdroj [8]). Paralelní souřadnice byly vynalezeny v roce 1885 vědcem Maurice d'Ocagnem. Tehdy nenašly využití a byly zapomenuty. Alfred Insenberg je roku 1959 znovuobjevil a od té doby nalezl spousty možností využití i mimo oblast zpracování obrazu.

Paralelní souřadnice jsou metodou určenou pro zpracování, ale i zobrazování vícerozměrných dat. Běžnou metodou pro zobrazení vícerozměrných dat je vykreslení do grafu pomocí Kartézských souřadnic. Tato metoda naráží na svá omezení již při zobrazování více než dvourozměrných dat. Paralelní souřadnice tento problém řeší.

Systém paralelních souřadnic reprezentuje jednotlivé rozměry prostoru jako rovnoběžné souřadné osy. Poloha v N-rozměrném prostoru je definována N-1 přímkami mezi těmito osami.



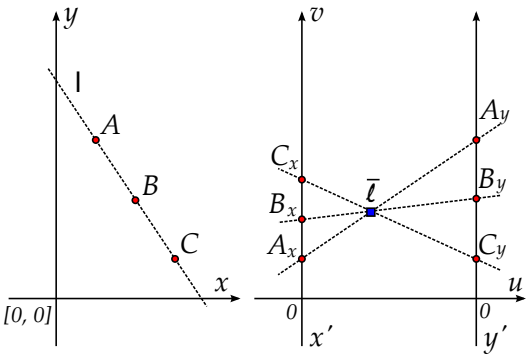
Obrázek 3.3: Vektor vložený do systému paralelních souřadnic. Převzato z [8].

Na obrázku 3.3 vidíme pětirozměrový prostor. Osy jsou pojmenovány $x'_1 \dots x'_5$. V daném prostoru se nachází vektor se souřadnicemi $[C_1, C_2, C_3, C_4, C_5]$. Vektor je zanesen tak, že na jednotlivých osách jsou vyznačeny hodnoty bodu v dané ose a následně jsou vytvořeny přímky mezi sousedními souřadnými osami. Pro popis polohy bodu pro homogenní souřadnice systému paralelních souřadnic se používá zápis $(u, v, w)_{\mathbb{P}^2}$.

Pro naše použití si vystačíme s dvourozměrným prostorem, proto se dále budeme zabývat pouze jím. V tomto specifickém případě platí dualita podobně jako v Houghově transformaci v části 3.1.3. Obrázek 3.4 znázorňuje transformaci Bodů A, B, a C do systému paralelních souřadnic. Body jsou kolineární v Eukleidovském prostoru. I zde jako v Houghově transformaci mají takové body transformované na přímky společný jeden průsečík.

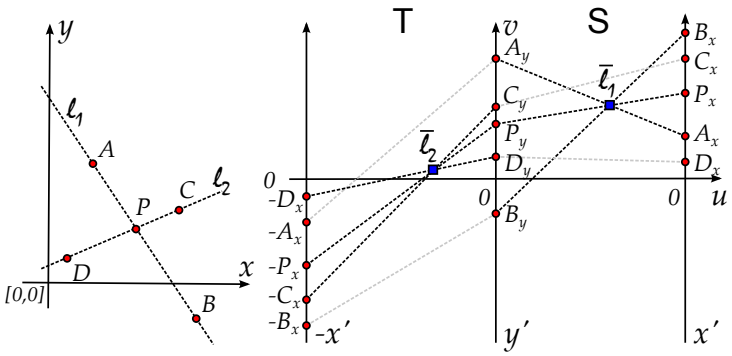
3.2.1 Detekce přímek

Mějme paralelní souřadnice x' a y' reprezentující Eukleidovské souřadnice x a y. V Eukleidovském prostoru repreznetuje přímku rovice $\ell : y = mx + b$. V paralelních souřadnicích je reprezentována trojicí $\bar{\ell} = (d, b, 1 - m)_{\mathbb{P}^2}$. d je vzdálenost mezi osami x' a y'. Dle [8] pak narazíme na omezení, takové, že ℓ je bod mezi x' a y' pouze tehdy když $-\infty < m < 0$. Toto



Obrázek 3.4: Transformace bodů na přímce. Převzato z [8].

omezuje možnost transformace přímek z Houghova prostoru pouze na přímky pod úhlem $90°a180°$ (vztaženo k ose x proti směru hodinových ručiček). Pro ostatní přímky v prostoru mezi x' a y' nevzniká společný průsečík. Podrobnosti jsou popsány v článku [8].



Obrázek 3.5: Ukázka transformace bodů dvou přímek do $\mathcal{TS}$ prostoru. Převzato z [8].

V [8] je popsáno i řešení problému. Přidáním další paralelní souřadnice $-y'$, nebo $-x'$ (jako na obrázku 3.5). Osa y' pak dělí prostor na dvě části. Část $y' - x'$ je nazývána přímá a označována $\mathcal{S}$ (straight). Nově vzniklá část $-x' - y'$ se nazývá převrácená s označením $\mathcal{T}$ (twisted). Převrácená část umožňuje transformovat přímky pod úhlem $0° - 90°$ pro které platí $0 < m < \infty$.

Kompletně celý prostor na obrázku 3.5 je označován jako $\mathcal{TS}$ prostor. Za povšimnutí také stojí změna souřadného systému z $x - y$ s počátkem v levém dolním rohu na $u - v$ se středem uprostřed $\mathcal{TS}$ prostoru. Pro $\mathcal{TS}$ prostor nyní platí následující omezení 3.5 a 3.6:

$$-d \leq u \leq d \tag{3.5}$$

$$-max\left(\frac{W}{2}, \frac{H}{2}\right) \leq v \leq max\left(\frac{W}{2}, \frac{H}{2}\right) \tag{3.6}$$

W a H reprezentují výšku a šířku Eukleidovského prostoru, tedy vstupního obrazu. Zajímavou vlastností $\mathcal{TS}$ prostoru je chování známé u Moebiovy pásky. Jestliže za počátek prostoru vezmeme osu $-x'$, pokračujeme přes y', dojdeme k x', po jejím převrácení jsme opět na začátku na ose $-x'$.

$\mathcal{TS}$ lze implementovat podobně jako Houghův prostor jako dvourozměrné pole akumulátorů. Přímky vytvářejí v $\mathcal{TS}$ prostoru charakteristické motýlky a lze je snadno detekovat jako lokální maxima. Polohy přímek ve vstupním obraze poté určíme pomocí $\theta - \rho$ parametrizace podle následujících vzorců:

$$\theta = \arctan\left(\frac{u}{d-u}\right) \tag{3.7}$$

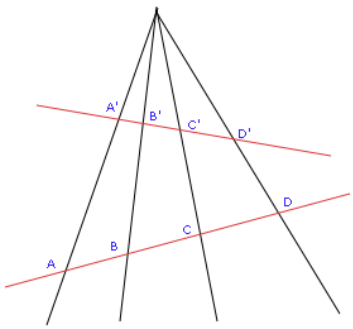
$$\theta = \arctan\left(\frac{u}{d+u}\right) \tag{3.8}$$

$$\rho = \frac{vd}{\sqrt{(d-u)^2 + u^2}} \tag{3.9}$$

V rovnicích 3.7, 3.8 a 3.9 je ekvivalentní stejná $\theta - \rho$ parametrizace jako u Houghovy transformace. Tedy θ je úhel sevřený s osou x a ρ nejkratší vzdálenost přímky od středu souřadného systému $u - v$. Rovnice 3.7 je platná pro zpětnou transformaci bodu z $\mathcal{T}$ podprostoru a 3.8 pro $\mathcal{S}$ podprostor.

3.3 Cross-ratio

Cross-ratio je dvojitý poměr, tedy jedno konkrétní číslo, popisující vzdálenosti mezi rovnoběžkami na které byla aplikována transformace perspektivní projekce.



Obrázek 3.6: Ilustrace situace rovnoběžek promítnutých perspektivní projekcí. Převzato z [2].

$$cross - ratio\,(A, B; C, D) = \frac{AC \cdot BD}{BC \cdot AD} \tag{3.10}$$

Mějme 4 rovnoběžky (jako na obr. 3.6). Provedeme jejich perspektivní projekci. Potom plati, že pro každou přímku vloženou tak, že vzniknou průsečíky se všemi čtyřmi rovnoběžkami, je dvojitý poměr vzdáleností mezi průsečíky popsaný v 3.10 stejný. Pro úplnost vysvětlení obrázku, platí:

$$cross - ratio\,(A, B; C, D) = cross - ratio\left(A', B'; C', D'\right)$$

Kapitola 4

Návrh vlastního algoritmu pro detekci maticového kódu

Kapitola se zabývá návrhem vlastního algoritmu na základě teorie uvedené v předcházející kapitole. Návrh algoritmu bere v úvahu požadavky uvedené v zadání, ale i další doplňující požadavky uvedené níže. V poslední části kapitoly jsou popsaná omezení plynoucí z návrhu algoritmu.

Vlastní algoritmus by měl splňovat tyto požadavky:

- využití PClines algoritmu pro detekci přímek v obraze
- rozpoznání a detekce maticového kódu v co možná nejkratším čase (v reálném čase)
- maximální přesnost detekce
- co nejnižší počet falešných detekcí kódu

Základní struktura algoritmu je následující:

1. detekce, kategorizace, filtrace hran
2. akumulace Houghova prostoru
3. detekce rovnoběžek - lokalizace v Houghově prostoru
4. extrakce maticového kódu z obrazu

Vlastní algoritmus by měl být dále co nejvíce univerzální ve směru detekce různých typů kódů. Pro potřeby vytvoření aplikace demonstrující detekci maticového kódu je jeho závěrečná část zaměřena na detekci QR kódů.

4.1 Extrakce hran

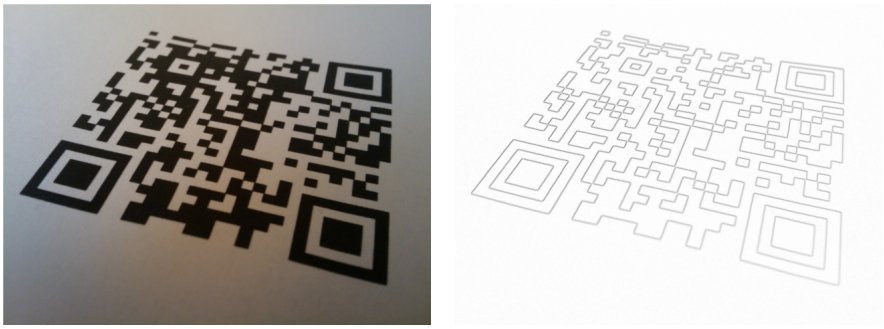
Prvním krokem algoritmu je detekce a filtrace hran, které jsou vstupem do algoritmu PClines. Volba hranového detektoru má zásadní vliv na vlastnosti algoritmu. Filtrace pomocí binů se snaží o výběr hranových bodů patřících rovnoběžným přímkám.

4.1.1 Detekce hran

Z mnoha různých detektorů hran se jeví Sobelův operátor jako vhodný kompromis mezi rychlostí a kvalitou odezvy na hranu. Operátor je relativně rychlý, pro získání gradientního obrázku je třeba pouze násobení a sčítání několika hodnot v okolí počítaného bodu. Nevýhodou je, že neostrá hrana vytvoří více odezev v gradientním obrázku a přesnost vypočítaného směru gradientu při užití malých jader nemusí být příliš vysoká.

Vhodou filtraci provádí Cannyho detektor. Je ale výpočetně náročnější a hrany z jeho výstupu by bylo také nutné filtrovat, aby bylo možné ovlivnit dobu následné akumulace Houghova prostoru.

Vstupem do Sobelova operátoru je šedotónový obraz. Definice barev užitých v QR kódu říká, že jedna barva by měla být světlá a druhá tmavá. Jiné maticové kódy mají barvy specifikované podobně. Předpokládám tedy, že by převod do odstínů šedi měl zanechat co nejvíce obrazové informace v oblasti maticového kódu. Proto není třeba řešit použití Sobelova operátoru nad barevným obrazem.



(a) Vstup (b) Sobelův operátor

Obrázek 4.1: Odezva Sobelova operátoru na vstupní obraz.

Ostré přechody mezi černou a bílo barvou v QR kódu na obrázku 4.1 dávají kvalitní odezvu hranového detektoru.

4.1.2 Kategorizace hran

Pro filtraci hran byl nejprve navrhnut vlastní algoritmus dále popsaný v 4.1.3. Jeho efektivita nebyla příliš vysoká, proto byla poté použita varianta algoritmu popsaná v článku [10].

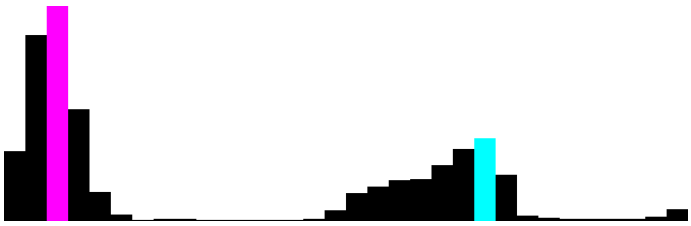
Předpokládám, že v ideálním případě hrany v maticovém kódu na obrázku tvoří dvě skupiny vzájemně kolmých rovnoběžek. Zkreslením perspektivní projekcí však dochází k tomu, že rovnoběžky v maticovém kódu nemusí být rovnoběžné, avšak sousedící dvě rovnoběžky mají vždy velmi podobný úhel. Hranové body kategorizujeme do tzv. binů odpovídajících určitým rozsahům směrů hran. Za předpokladu, že sady rovnoběžek v obraze mají dostatečně odlišitelné úhly, je možné přiřadit body z jednotlivých binů k sadám rovnoběžek.

Přímky mohou nabývat úhlů v rozsahu $0°$ až $360°$. Směr hrany před vložením do binů je vhodné normalizovat do rozsahu od $-90°$ do $90°$. Ostatní směry hran převedeme na ekvivalentní otočením o $180°$. Potom prostor $-90°$ až $90°$ rozdělíme do N stejně velkých rozsahů. Každý rozsah je přidělen jednomu binu. Při průchodu gradientními obrázky pro každý bod, který má větší odezvu, než určitý experimentálně určený práh, přičteme velikost odezvy na hranu k patřičnému binu. Velikost odezvy se vypočítá pomocí Pythagorovy věty.

Směr hrany v daném bodu vypočítáme pomocí rovnice 4.1, kde s_x a s_y reprezentují velikost odezvy Sobelova operátoru v daném bodě ve směru osy x a y. Výsledný úhel se vztahuje ke kladné části svislé osy, proti směru hodinových ručiček.

$$\alpha = -atan\left(\frac{s_x}{s_y}\right); \tag{4.1}$$

Výslednou odezvu na maticový kód můžeme vidět na obrázku 4.2. Z obrázku je zřejmé, že lze snadno najít dva dominantní směry hran označené barevně. Vrcholy v okolí vybraných binů reprezentují sady rovnoběžek.



Obrázek 4.2: Velikosti jednotlivých binů, 2 dominantní směry barevně.

4.1.3 Filtrace hran

Po nalezení dominantních směrů je možné pro každou sadu rovnoběžek vymezit rozsah jejich úhlů. Toto lze provádět například tak, že k binu s dominantním směrem postupně přidáváme okolní biny do té doby, než je velikost binu menší, než určitá část velikosti dominantního binu. Nyní máme všechny směry hran rozděleny na ty náležící jedné, nebo druhé sadě rovnoběžek a na ty, které mají jiný směr.

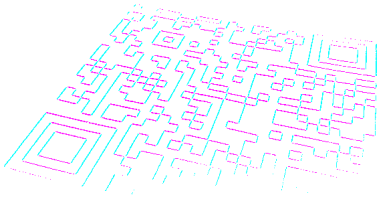
První navrhovaný filtr rozdělil obraz na malé obdélníky. Např. 20 x 20. V každé z těchto oblastí byl vypočítáván poměr bodů náležících hranám rovnoběžek a ostatních bodů. Předpokládal jsem, že oblasti uvnitř maticového kódu budou splňovat nějaký minimální poměr, podle kterého by bylo možné rozdělit oblasti na ty, které obsahují maticový kód a které nikoliv. Právě nalezení prahu pro minimální poměr byl problém tohoto filtru. Nebylo možné nastavit práh, který bý byl univerzální pro celou, nebo dostatečně velkou část testovací sady. Vhodný minimální poměr byl pro každý obraz individuální a proto bylo od tohoto filtru upuštěno.

Jako náhrada je použit filtr popisovaný v [10]. Biny máme tedy rozděleny podle sad rovnoběžek. Potom pro každou sadu nalezneme E hranových bodů takových, že ve své sadě mají největší odezvu na hranu a současně je tato odevza vyšší, než minimální požadovaná odezva. Tyto body postupují dále do akumulace Houghova prostoru.

Na obrázku 4.3 vidíme, že takový filtr propouští výrazné hrany dobře popisující maticový kód a zahazuje hrany slabší (rozmazané, méně kontrastní).

4.2 Detekce rovnoběžek

Výhodou užití PClines oproti Houghově transformaci, ale i jiným algoritmům pro detekci maticových kódů, je velmi snadná detekce rovnoběžek. Rovnoběžné přímky ve vstupním obraze vytvářejí v $\mathcal{TS}$ prostoru řadu motýlků, tedy bodů lokálních maxim, které jsou téměř kolineární. Toto závisí hlavně na rozlišovací schopnosti implementace $\mathcal{TS}$ prostoru a množství šumu ve stupním obraze, který může ovlivnit přesnost nalezené polohy lokálních maxim.



Obrázek 4.3: Odezva filtru na hrany. První sada rovnoběžek modře, druhá růžově.

4.2.1 Nalezení výrazných přímek

Aby lokální maxima identifikující přímky byla vzájemně kolienární, je třeba pro každou sadu rovnoběžek zvolit způsob akumulace prostoru jako $\mathcal{T}$, nebo $\mathcal{S}$. Toto zvolíme podle dominantního binu. Houghův prostor začíná úhlem $-180°$ a končí úhlem $0°$ pro prostor typu $\mathcal{T}$. $\mathcal{S}$ prostor začíná úhlem $0°$ a končí úhlem $180°$. Dominantní bin spadá zcela jistě do jedné z těchto kategorií.

Nyní je vytvořen pro každou sadu rovnoběžek jeden Houghův prostor. Alokace probíhá tak, že pro každý bod hrany, který byl propuštěn filtrem, vytvoříme úsečku začínající v jedné ose a končící v jiné ose (princip popsaný zde 3.2.1). Do $\mathcal{TS}$ prostoru není třeba přímku vykreslovat celou. Z binů ohraničujících danou sadu rovnoběžek vypočítáme krajní úhly a určíme její počátek a konec v $\mathcal{TS}$ prostoru ve vodorovném směru. Inkrementujeme pouze akumulátory odpovídající přímce mezi těmito hranicemi.

Následuje nalezení nejvýraznějších přímek. Ty nalezneme jako lokální maxima $\mathcal{TS}$ prostoru. Vidíte na obrázku 4.4.

4.2.2 Nalezení rovnoběžných přímek v Houghově prostoru

K nalezení přímky protínající všechna lokální maxima, tj. vedoucí skrze všechny body $\mathcal{TS}$ prostoru reprezentující rovnoběžky ve vstupním obraze, použijeme opět PClines algoritmus.

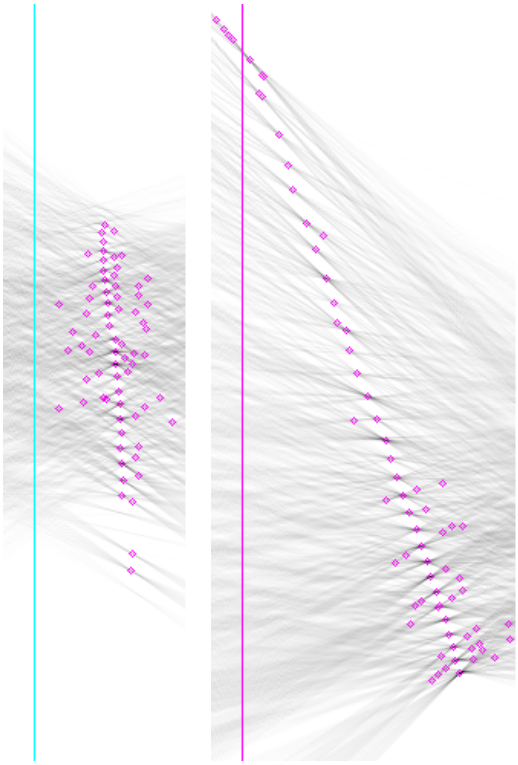
Vytvoříme nový Houghův prostor složený z $\mathcal{T}$ a $\mathcal{S}$ částí. Vložíme do něj nalezená lokální maxima. Nyní hledáme jedno lokální maximum reprezentující hledanou přímku v prvním $\mathcal{TS}$ prostoru. Jak je vidět na obrázku 4.5, vložené body - přímky vytvoří zřetelné lokální maximum.

4.2.3 Nalezení kandidátů rovnoběžných přímek

Nyní se vrátíme zpět k prvnímu Houghovu prostoru. Po nalezení přímky, procházející skrze lokální maxima, provedeme podle ní řez Houghovým prostorem. Získáme tak vektor hodnot znázorněný na obrázku 4.6.

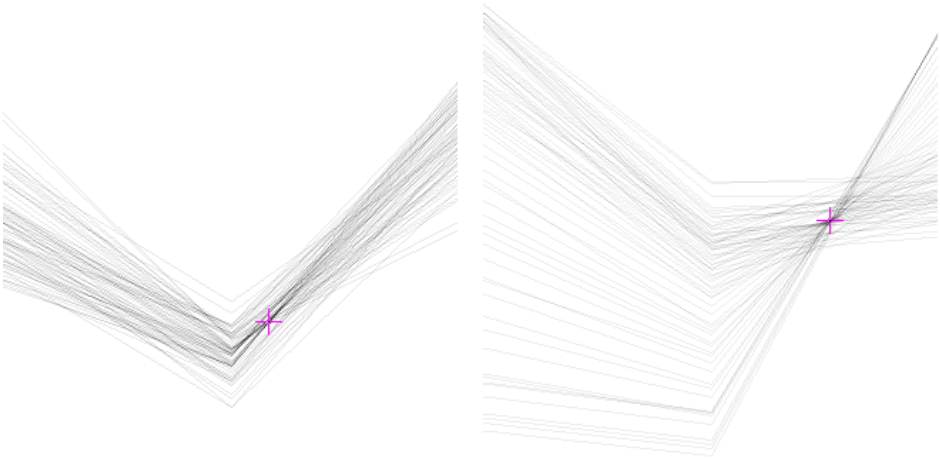
Z tohoto vektoru jsme okem schopni rozeznat jednotlivá lokální maxima reprezentující rovnoběžky. Problémem identifikace lokálního maxima je ale určení velikosti lokality. Jednotlivá lokální maxima mohou být od sebe různě vzdálená vlivem zkreslení maticového kódu při perspektivní projekci. Proto nelze určit univerzální velikost lokality.

Proto zde zvolíme jiný algoritmus. Špička je definovaná jako bod s maximální hodnotou, poté co profil klesl pod prahovou hodnotu. Prahová hodnota se určí jako malá část maximální hodnoty. Tímto způsobem se podaří najít všechny výrazné špičky a odfiltrovat



(a) sada 1 (b) sada 2

Obrázek 4.4: Ukázka nalezených lokálních maxim v $\mathcal{TS}$ prostoru.

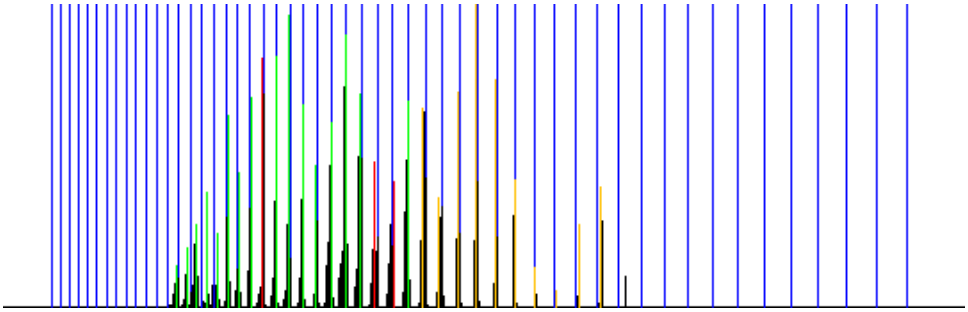


(a) sada 1 (b) sada 2

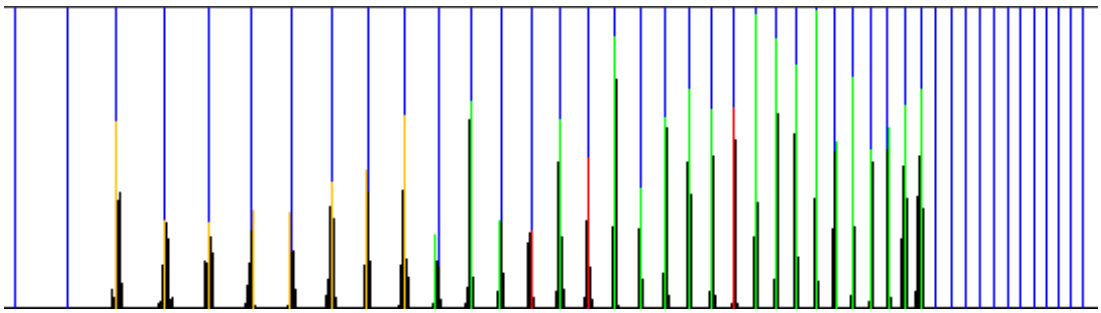
Obrázek 4.5: Lokální maxima vložená do druhého $\mathcal{TS}$ prostoru.

některé špičky, které jsou šumem. Z vybraných špiček, s vyjímkou nejvyšší a nejnižší hodnoty, se vypočítá průměrná výška. Z určité části této výšky se vytvoří nový práh podle kterého jsou špičky ještě jenou profiltrovány.

Výsledkem by měla být sada špiček, ve které mají všechny špičky podobnou výšku. Filtrací špiček jsou totiž odstraňovány drobné špičky detekované v šumu, který v profilu



(a) sada 1



(b) sada 2

Obrázek 4.6: Profily získané řezem $\mathcal{TS}$ prostoru na obrázku 4.5. černě - profil, žlutě - detekované špičky na profilu, které nejsou inliers, zeleně - špičky - inliers, červeně - počáteční špičky ze kterých se vygeneroval model, modře - nalezený model.

vznikne okolo maticového kódu vlivem nedokonalé filtrace hranových bodů v okolí maticového kódu.

4.2.4 Nalezení modelu sady rovnoběžek

Tato část algoritmu se zaměřuje na vytvoření vypočítaného modelu sady rovnoběžek, který bude nejlépe popisovat polohu a rozložení matocového kódu v obraze. Něž začneme takový model vytvářet potřebujeme získat trojici rovnoběžek, kterými se bude model inicializovat. Potřebujeme co nejspolehlivěji zaručit, že tato trojice rovnoběžek (špiček) a rovnoběžky mezi nimi patří do dané sady rovnoběžek a že se mezi nimi nenachází žádná falešná přímka.

V této části je celý problém až na výpočet metriky redukován na jednorozměrný. Každá špička je zde reprezentována vzdáleností od počátku profilu.

Hledáme co nejdelší posloupnost špiček takovou, že každá z nich reprezentuje jednu z rovnoběžek. Tato posloupnost se nazývá inliers. Ostatní špičky jsou potom outliers.

Využijeme zde znalost Cross-ratio. Náhodně volíme trojice špiček. Těmito trojicemi se snažíme verifikovat čtvrtou špičku - ověřujeme zda patři do stejné sady rovnoběžek jako náhodná trojice. Počety špiček mezi čtyřmi již uvedenými (1 pro sousední špičky, 2 pro špičky přes jednu…) vložíme do vzorečku jako vzdálenosti pro výpočet Cross-ratio 3.10. Poté do stejného vzorce vložíme dvě vzdálenosti vypočítané z profilu mezi trojicí nahodně vzbraných špiček a vypočítané Cross-ratio. Získáme tak vzdálenost ve které by se v ideálním případě měla nacházet verifikovaná špička. Porovnáme vzdálenost se skutečnou vzdáleností a podle velikosti rozdílu a prahu vyhodnotíme zda je špička rovnoběžkou s dalšími třemi, či nikoliv.

Takovýchto trojic je třeba otestovat několik z níže popsaných důvodů. Ze všech testovaných trojic vybereme co nejdelší posloupnost inliers.

Při verifikaci náhodnou trojicí mohou nastat 3 situace:

1. jedna špička z trojice není rovnoběžkou - v takovém případě se touto trojicí nepodaří verifikovat většinu ostatních špiček, nebo žádnou

2. mezi špičkami náhodně generované trojice chybí špička reprezentující rovnoběžku, nebo přebývá špička, která není rovnoběžkou - v takovém případě se pokaždé chybně vypočítá Cross-ratio, vzdálenosti budou vypočítávány chybně a opět se nepodaří verifikovat téměř žádné, nebo žádné špičky

3. všechny tři špičky jsou rovnoběžky a počty špiček mezi nimi odpovídají počtu rovnoběžek, který by mezi nimi měl být. (Může nastat situace, kdy počet špiček odpovídá, ale špičky nejsou na správných místech) - pouze v tomto případě je možné nalézt co nejdelší posloupnost špiček reprezentujících rovnoběžky

Nyní můžeme přistoupit k části vytváření modelu. Pohybujeme se v jednorozměrném prostoru na přímce v Houghově prostoru. Tuto přímku si nyní omezíme na úsečku tak, že ji ohraničíme výškou Houghova prostoru. Jednu hranici si označíme jako začátek a druhou jako konec. Potom může být každá sada špiček reprezentujících rovnoběžky, ležících na této přímce, reprezentována trojicí indexů i_1, i_2, i_3 náležících trojici rovnoběžek reprezentovaných body x_1, x_2, x_3. Polohy těchto bodů lze přesně zapsat vzdálenostmi d_1, d_2, d_3 od nějakého bodu na úsečce. Jako tento bod si zvolíme bod x_s, začátek úsečky(profilu). Z indexů a vzdáleností rovnoběžek jsme schopni vypočítat všechny ostatní body v Houghově prostoru náležící ostatním rovnoběžkám. Tyto dva údaje tedy plně postačují k vytvoření modelu rovnoběžek nad celým profilem.

Pro dokončení modelu inicializovaného trojicí špiček, už zbývá pouze dopočítat ostatní body reprezentující ostatní rovnoběžky. Toto provedeme obdobným způsobem jako u verifikace špiček. Dosadíme do Cross-ratio rovnice 3.10 počty rovnoběžek jako vzdálenosti, vypočítáme Cross-ratio. Dosadíme do Cross-ratio rovnice skutečné vzdálenosti mezi body x_1, x_2, x_3 a vypočítané Cross-ratio, získáme vzdálenost čtvrté rovnoběžky. Podobným způsobem dopočítáme dostatečné množství dalších rovnoběžek.

O množství rovnoběžek na profilu napovídá množství detekovaných špiček. Toto číslo většinou není přesné, je bráno jako základ a je k němu z obou stran vygenerováno několik (5-15) dalších rovnoběžek, více v kapitole 6.

Kvalitu vygenerovaného modelu je třeba nějakým způsobem ohodnotit. Pro tyto potřeby si k rovnoběžkám reprezentujícím hrany v maticovém kódu vygenerujeme i rovnoběžky protínající středy modulů maticových kódů. Metriku M pro výpočet přesnosti modelu vypočítáme dle vzorce 4.2.

$$M = M_p + M_v \tag{4.2}$$

$$M_p = \sum_{i=0}^{i=m} -p(i) \tag{4.3}$$

$$M_v = \sum_{j=0}^{j=n} p(j) \tag{4.4}$$

Máme m rovnoběžek popisujících hrany, je pro ně vygenerováno $n = m - 1$ rovnoběžek protínající středy modulů tj. údolí v profilu. Metrika M_p počítá chybu všech generovaných špiček jako zápornou hodnotu součtu všech hodnot profilu v jejich místech $p(i)$. Metrika M_u počítá chybu v bodech ve kterých by se mělo vyskytovat údolí, tedy optimálně nulová výška profilu. Výsledkem je součet všech hodnot profilu v bodech vypočítaných jako údolí $p(j)$. Obě části metriky M jsou výpočet velikosti chyby. V ideálním případě budou jejich hodnoty co nejmenší, potom i hodnota metriky M by měla být co nejmenší.

Takto vytvořený model bude přesně pasovat v třech náhodně vybraných vrcholech, ze kterých byl generován. Se vzdáleností generovaných rovnoběžek od generující trojice klesá přesnost modelu a model přestává odpovídat profilu ze kterého byl vytvořen.

Proto je nutné model upravit. Úprava je jednoduchá. Posuneme bod x_1 o vhodně zvolenou vzdálenost (index bodu zůstává stejný) e_0 směrem k počátku profilu, vygenerujeme nový model a vypočítáme hodnotu metriky. Získáme tak hodhotu M_l. Totéž provedeme s posunem na opačnou stranu z původní pozice a získáme M_r. Pokud je hodnota metriky ve výchozím bodu nejnižší, v této fázi s ním již nehýbeme. Jinak nalezeme nejnižší hodnotu metriky z posunutých bodů a zkusíme posunout tímto směrem znovu o vzdálenost e_0. Toto děláme dokud výsledek metriky klesá. Až se poloha bodu x_1 ustálí, pokračujeme stejným způsobem s body x_2 a x_3. Poté zmenšíme vzdálenost e na polovinu $e1 = e_0/2$ a opakujeme algoritmus posouvání s novou hodnotou. Hodnotu e je možné takto několikrát zmenšit a dostat se tak od hrubého posunu k jemnějšímu a vytvořit tak model, který co nejlépe odpovídá hodnotám v profilu.

Vzhledem k šumu v profilu model v některých případech nedoiteruje k ideálnímu modelu. Tzn. algoritmus uvízne v lokálním minimu funkce metriky. Tomuto by měla bránit zmenšující se velikost, vzdálenosti e. V takovém případě by ale všechny modely měly doiterovat ke stejnému stavu. Testy však ukazují, že tomu tak není. Výsledek je závislý na trojici bodů, kterou model inicializujeme. Proto vytváříme několik modelů inicializovaných různými náhodně zvolenými trojicemi z inliers. Z těchto modelů je potom vybrán model s nejnižší výslednou metrikou.

Hlavním zájmem této části algoritmu jsou údolní body na profilu získané z nejlepšího modelu. Vzhledem k tomu, že známe počátek i konec přímky v profilu, není problém vypočítat jejich uv souřadnice v $\mathcal{TS}$ prostoru. Tyto body jsou poté převedeny rovnicemi popsanými v 3.2.1 na přímky v θ - ρ parametrizaci. Jsou to ty přímky, které procházejí středy modulů.

4.2.5 Verifikace špiček - méně úspěšná varianta algoritmu

První navrhovaná varianta této části algoritmu byla mnohem jednodušší, avšak nedosahovala příliš uspokojivých výsledků a proto od ní bylo v průběhu vývoje upuštěno. Výsledky úspěšnosti rozpoznávání jsou zahrnuty v tabulce 6.2 v části 6.3.

Metoda je shodná s předchozí popsanou do stavu, kdy máme detekované špičky na profilu a z nich vybrané inliers.

Následuje fáze verifikace špiček. Nad inliers je náhodně vybrána trojice špiček a pomocí této trojice jsou vypočítávány pozice dalších špiček na každou stranu v okolí inliers. Začíná se jednou z inicializačních špiček. Pokud je rozdíl poloh těchto vypočítaných špiček a detekovaných špiček větší než práh, je do výsledných bodů vložen bod vypočítaný, jinak se vkládá bod detekovaný jako špička. Po vložení určitého počtu vypočítaných špiček po sobě přidávání bodů tímto směrem od prvního verifikovaného bodu končí. Stejně tak pokud má být přidána špička mimo $\mathcal{TS}$ prostor. Pokračuje se na druhou stranu od tohoto bodu.

Vybrané body, u kterých je to možné, jsou dále v $\mathcal{TS}$ prostoru posunuty směrem k nejbližším lokálním maximům metodou hill climbing.

Nakonec jsou mezi všemi sousedními úspěšně verifikovanými body vypočítány středy. Tyto středy jsou body reprezentující přímky procházející středy modulů maticových kódů. Tyto body postupují do části extrakce obrazu.

4.3 Extrakce maticového kódu z obrazu

Z předchozích částí algoritmu máme k dispozici šedotónový vstupní obraz a dvě sady rovnoběžných přímek v θ - ρ parametrizaci.

Průsečík dvou přímek s parametry θ_1, ρ_1, θ_2, ρ_2 potom vypočítáme dle rovnic 4.5 a 4.6 pro $\theta_1 < \pi$ a 4.8 a 4.7 pro ostatní θ_1. Toto rozdělení má za následek vyšší přesnost při výpočtu pro θ_1 blízké 0, nebo $\pi/2$. Dále se tak vyhneme problému s dělení nulou.

$$v = \frac{\rho_1 * cos(\theta_2) - cos(\theta_1) * \rho_2}{sin(\theta_1) * cos(\theta_2) - sin(\theta_2) * cos(\theta_1)} \tag{4.5}$$

$$u = \frac{\rho_1 - v * sin(\theta_1)}{cos(\theta_1)} \tag{4.6}$$

$$u = \frac{\rho_1 * sin(\theta_2) - sin(\theta_1) * \rho_2}{cos(\theta_1) * sin(\theta_2) - cos(\theta_2) * sin(\theta_1)} \tag{4.7}$$

$$v = \frac{\rho_1 - u * cos(\theta_1)}{sin(\theta_1)} \tag{4.8}$$

Rovnice vycházejí z rovnice 3.3. Vzniknou dosazením rovnice jedné přímky do druhé. Důležité je nezapomenout, že výsledkem jsou u-v souřadnice, které mají střed uprostřed vstupního obrazu.

Výše popsaným způsobem vypočítáme všechny průsečíky všech přímek z první sady se všemi přímkami ze sady druhé. V bodech vypočítaných průsečíků přečteme pixel z šedotónového obrázku. Tyto body jsou znázorněny na obrázku 4.7. Vzhledem k tomu, že sady rovnoběžek obsahovaly nekonečně dlouhé přímky, některé průsečíky mohou ležet mimo hranice vstupního obrazu. Hodnoty takových průsečíků doplníme na univerzální, neutrální hodhotu. Tj.prostřední hodnotu šedotónové stupnice (128 pro rozsah stupnice šedi 0 - 255). Výslednoý obraz můžeme vidět na obrázku 4.8

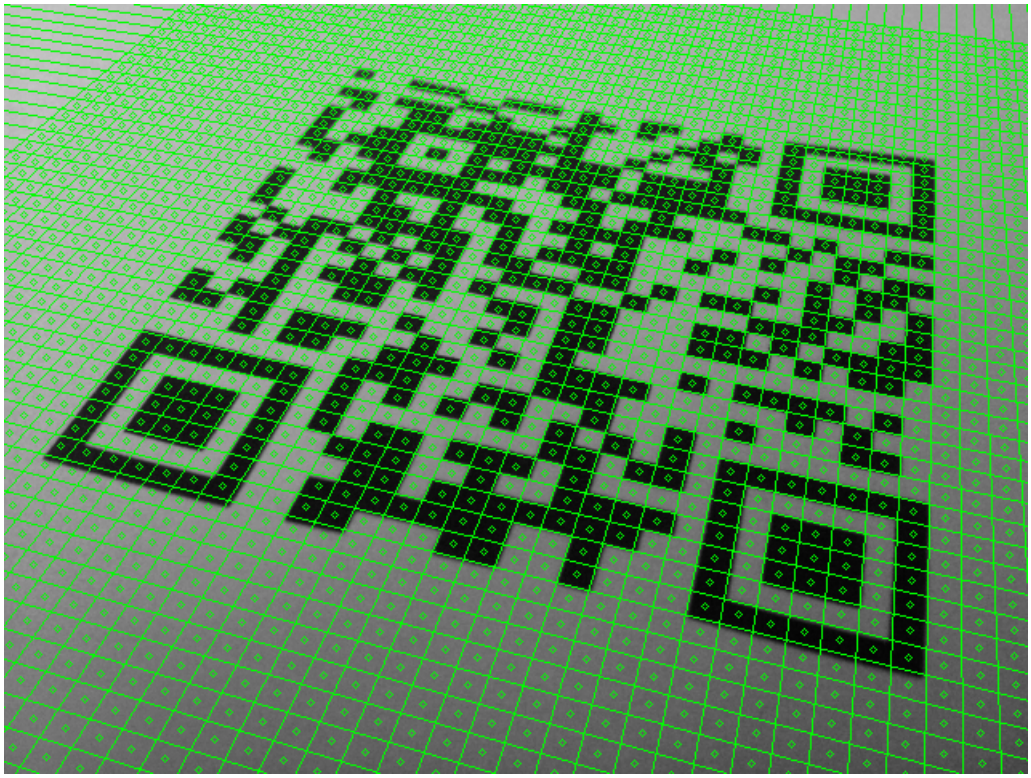
Černobílou bitmapu, která je vidět na obrázku 4.9 získáme lokálním prahováním šedotónového extraktu z předchozí části. Lokální prahování by mělo dosahovat lepších výsledků, než globální práh, protože maticový obraz nemusí být rovnoměrně osvětlen.

Struktura zde reprezentovaná černobílou bitmapou na obrázku 4.9 je výsledkem navrhnutého algoritmu pro detekci maticových kódů.

4.4 Detekce QR kódu v bitmapě

Algoritmus je až do tohoto bodu univerzální ve směru rozpoznání různých typů maticových kódů. Další část je zaměřena čistě na detekci QR-kódů.

QR kód je v bitmapě z předchozího kroku identifikován pomocí jeho rohových značek. Nejprve je nalezena rohová značka s co nejlepší odezvou, tj. ideálně bez chyby. Je prováděn template matching. Následně jsou hledány další rohové značky ve všech osmi směrech a to



Obrázek 4.7: Rovnoběžky reprezentující hrany maticového kódu a vypočítané středy modulů(kolečka).

Obrázek 4.8: Šedotónový extrakt z průsečíků a šedotónového obrazu.

ve vzdálenostech, v jakých by se mohly vyskytovat v QR kódu. Tato vzdálenost se počítá z možných velikostí QR kódů(21, 25, 29 ... modulů) a velikosti rohové značky (7 modulů). Při každém kroku vzdálenosti zjistíme nejlepší odezvu tří rohových značek(víme, které vzájemné polohy rohových značek maticového kódu jsou jeho rotace a které nikoliv). tuto trojici je třeba si pamatovat. Hledáme nejlepší odezvu trojice rohových značek přes všechny vzdálenosti a všechny rotace.

Ze vzájemné polohy značek v bitmapě je rozpoznána rotace kódu a díky tomuto je správně přečten. Výsledek takto zpracované bitmapy je vidět na obrázku 4.9.

Obrázek 4.9: Extrahovaná bitmapa.

Obrázek 4.10: Bitmapa detekovaného QR kódu.

4.5 Mechanismus zabraňující falešné detekci kódu

Aby bylo zabráněno snaze o detekci maticového kódu v obraze, ve kterém žádný maticový kód není, je vhodné do algoritmu zavést další mechanismy. Tyto mechanismy musejí být dostatečně jednoduché, aby jejich vliv na výpočetní čas byl co nejmenší.

1. mechanismus zabraňuje snaze o detekci na obraze s nedostatečně výraznými hranami. Pokud všechny odezvy Sobelova operátoru v obraze nepřesáhnou určitý práh, detekce je ukončena. Pro obraz s maticovým kódem se předpokládá, že budou hrany dostatečně ostré a kontrastní.

2. mechanismus předpokládá, že bude v každém profilu detekováno alespoň X špiček. Přičemž $X > 3$, aby bylo možné z těchto 3 špiček inicializovat a vytvořit model. Vysoké X zvyšuje pravděpodobnost zahození platného maticového kódu, avšak pro reálné použití aplikace je nutné najít správnou hodnotu.

3. mechanismus je záležitostí extrakce konkrétního maticového kódu z černobílé bitmapy. například u maticového kódu je možné kontrolovat kvalitu extrakce rohových značek při porovnání s vzorovými značkami. Poté je možné najít vhodnou minimální procentuální hodnotu shody se vzorem pro pokračování, či zamítnutí detekce.

4.6 Omezení navrhnutého algoritmu

Algoritmus byl navrhován co nejrobustnější a pro co nejširší paletu maticových kódů. I přes tyto požadavky lze z principu funkčnosti jeho jednotlivých částí (PClines, Cross-raio) odvodit jistá omezení:

- lze detekovat pouze maticové kódy které mají čtvercovou síť - např. úprava pro obdélníkovou síť by nebyla náročná, ale úprava pro síť šestiúhelníků, trojúhelníků, nebo kruhů by byla náročná, zahrnovala by zásadní změny
- síť maticového kódu musí mít všechny moduly stejně velké - existují maticové kódy, které v sobě mají vloženy zmenšeniny samy sebe (vytvářejí fraktály do určité hloubky) - úprava v tomto směru by měla být možná
- algoritmus je silně závislý na detekci hran - z rozmazaného obrazu bude obtížné detekovat maticový kód
- algoritmus je citlivý na zvlněnou plochu na které se kód nachází, nebo na zkreslení obrazu vlivem kulatosti čočky - Cross-ratio rovnice s těmito deformacemi nepočítá
- chování algoritmu při více maticových kódech na jednom obraze je nepředvídatelné- obě sady rovnoběžek budou detekovány na jednom maticovém kódu. Bude tedy detekován maticový kód s ostřejšími hranami. Nebo bude každá sada z jiného maticového kódu. Výsledná bitmapa bude obsahovat body extrahované v průsečících těchto sad rovnoběžek

Kapitola 5

Implementace

Aplikace byla implementována v jazyce C++ s použitím standartních knihoven. Tento jazyk byl zvolen s ohledem na programátorské zkušenosti, rychlost kompilovaného programu a multiplatformnost. Pro operace načítání obrazu, zobrazování výsledků a další obrazové operace, byla využita knihovna OpenCV. Implementace je univerzální ve směru detekce libovolného maticového kódu. Rozpoznání QR kódu je volitelná část implementovaná třídou `QRExtractor`.

V souboru `main.cpp` se nachází několik funkcí `main(...)`. Funkce se spustí v závislosti na přepínači podmíněného překladu. Jedna z funkcí spouští test prostřednictvím testovací třídy `Tester`. Třída provede test nad celou datovou sadou a vypíše výsledky. Jiná funkce `main(...)` je přeložena s parametry pro ladění a zobrazuje množství grafických výstupů sloužících k ladění chyb. Tuto možnost překladu je možné vnímat zároveň i jako demonstrační. Popis překladu a použití programu je popsáno v příloze.

5.1 Detekce hran a klasifikace jejich směru

Pro detekci hran v obraze je využíván Sobelův operátor z knihovny OpenCV. Funkce je volána `cvSobel(gImg, sobX16S, 1, 0, 1);`. Při tomto volání je využíváno konvoluční jádro o velikosti 3x1 a výpočet probíhá ve vodorovném směru. Pro výpočet derivace v ose y je volání a chování funkce analogické. Jádro o velikosti 3x1 má oproti jádru 3x3 výhodu, jak ve zrychlení programu, tak v mírném zvýšení přesnosti výpočtu úhlů hran.

Směry hran jsou vypočítávány funkcí `atan2(y,x)`, která počítá arcus tangens z y/x. Velikost odezvy hrany je vypočítána Pythagorovou větou tedy `sqrt(Gy*Gy + Gx*Gx)`. Toto je sice nejpřesnější způsob výpočtu, avšak není nejrychlejší. Ryhchlejší způsob například aproximace součtem gradientů se jeví jako příliš nepřesný.

Hrany jsou filtrovány ve třídě `DirectBins`. Jsou filtrovány podle minimální délky a dále jsou rozdělovány do 32 binů. Bin není inkrementován o 1 ve chvíli kdy do něj hranový bod spadá, ale je k němu přičítána velikost odezvy hrany. Takto získávají ostřejší hrany výraznější vliv. U maticových kódů předpokládám výrazné hrany oproti okolí.

Dva největší biny jsou vybírány jako lokální maxima (pro 32 binů velikost okolí 13). Z procentuální části jejich velikosti je určena prahová hodnota. Pomocí prahové hodnoty jsou přiřazovány pro každý největší bin jeho okolní biny, dokud splňují podmínku velikosti větší, než práh. Takto jsou vytvořeny úhlové rozsahy pro dvě sady rovnoběžek. Do Houghova prostoru postupují takové hranové body, které splňují minimální délku a jejichž úhel je v rozsahu úhlů dané sady rovnoběžek.

Takto vznikne množina bodů pro danou sadu rovnoběžek a tato množina bodů je poté omezena na N (např. 5000) bodů s největší hranovou odezvou. To je prováděno tak, že při prvím průchodu této množiny je vytvořen histogram velikostí hranových odezev. Poté je z tohoto histogramu vybrán takový práh velikosti odezvy, aby se nad prahem v histogramu vyskytovalo maximálně N odezev. Je také uložena hodnota, kolik bodů nad prahem chybí do čísla N, označme si je C. Nad danou množinou bodů je poté proveden další průchod, kde jsou body přefiltrovány pomocí vytvořeného prahu tak, že projdou jen body s větší hranovou odezvou a prvních C bodů, které mají velikost odezvy shodnou s prahem.

5.2 Houghův prostor

Houghův prostor je implementován ve třídě `TSSpace` jako dynamicky alokované pole unsigned integerů. Bylo experimentováno s typem unsigned short, tento však při akumulaci Houghova prostoru nestačí.

Inkrementace akumulátorů v Houghově prostoru je z pohledu časové náročnosti kritická část. Algoritmem pro inkrementaci přímky akumulátorů mezi dvěma body je Bresenhamův algoritmus. Byl zvolen z důvodu jeho rychlosti, díky vyhýbání se výpočtům v plovoucí desetinné čárce. Přímka je vykreslována pouze v oblasti vymezené hraničními úhly pro danou sadu rovnoběžek. Vložením těchto úhlů do rovnic 3.7, nebo 3.8 vypočítáme omezení ve vodorovné ose u.

Kandidáty na přímky vyhledáme jako určitý počet (např. 50) největších lokálních maxim. Hlednání E největších maxim je výhodné v tom, že můžeme vzít prvních E bodů $\mathcal{TS}$ prostoru, seřadit je od největšího po nejmenší a poslední z nich určit jako práh. Při prohledávání prostoru potom kontrolujeme nejprve zda je daný bod větší než práh a až poté zda je lokálním maximem. Pokud je větší a je lokálním maximem, vložíme jej do E nejvyšších maxim, tak aby maxima zůstala seřazená. Poslední maximum tak bude vyloučeno z E nejvyšších a nové poslední maximum je opět prahem. Díky tomuto postupu práh relativně rychle roste a díky tomu se provádí méně porovnánvání při zjišťování, zda je daný bod lokálním maximem a celé prohledávání je tak rychlejší.

5.3 2. Houghův prostor

Ve třídě `TSSpace2` jsou body vloženy do nového $\mathcal{TS}$ prostoru. Tento prostor by mohl být implementován jako dvourozměrné pole typu char, ale z důvodu využívání některých metod ze třídy `TSSpace`, bylo ponecháno typu unsigned integer. V tomto prostoru není třeba hledat lokální maximum, jelikož hledáme pouze jeden bod. Dále byl tento prostor pomocí rovnic 3.7 a 3.8 omezen ve vodorovné ose na část $-45°$ až $45°$. Toto si lze dovolit, jelikož hledaná přímka v prvním $\mathcal{TS}$ prostoru skrze body reprezentující rovnoběžky by neměla být mimo tento rozsah. Znamenalo by to, že rozsah úhlů rovnoběžek ve vstupním obraze by byl větší, než $90°$. V takto deformovaném obraze by bylo značně obtížné hledat dvě různé sady rovnoběžek.

5.4 Detekce rovnoběžných přímek

Nalezli jsme přímku vedoucí skrze první $\mathcal{TS}$ prostor přibližně v místech dříve nalezených lokálních maxim. Ve třídě `TSLine` je získán profil této přímky pomocí Bresenhamova al-

goritmu. Následně jsou v této třídě identifikovány špičky. Špičky jsou verifikovány a jsou vytvářeny modely.

Tento postup často využívá výpočet rovnice cross-ratio, kde jsou koeficienty celá čísla. Předpokládám, že by bylo možné dosáhnout mírného zrychlení, pokud by byly výsledky pro všechny možné vstupy do nějaké maximální hodnoty předpočítány.

5.5 Extrakce maticového kódu

Z modelů jou získány přímky procházející středy modulů. Máme 2 sady těchto přímek. Jejich průsečíky vypočítáme jako průměr parametrů obou přímek (θ, ρ).

Po získání dvoubarevné bitmapy z matice vybraných pixelů (průsečíků) z obrázku, je použita funkce `cvAdaptiveThreshold(...)` z knihovny OpenCV. Způsob výpočtu prahu je zvolený parametrem `CV_ADAPTIVE_THRESH_GAUSSIAN_C`, což je průměr vážený Gaussovou funkcí.

Třída `QRExtractor` je dále připravena pro detekci QR kódu. Tato funkčnost není využívána při překladu programu pro testovací účely. Implementace kopíruje popis algoritmu v části 4.4.

Kapitola 6

Experimentální výsledky

Kapitola diskutuje nastavení různých proměnných v algoritmu, výslednou přesnost rozpoznávání a detekce a časovou náročnost výpočtů v jednotlivých částech algoritmu. Dále kapitola shrnuje dosažené výsledky a porovnává je s výsledky podobného algoritmu popsaného v článku [10].

Mnou užívaná testovací datová sada byla pořízena M. Dubskou a obsahuje 430 obrázků. Část obrázků byla pořízena kvalitním fotoaparátem, ostatní obrázky byly pořízeny různými mobilními telefony. Datová sada je oanotována, ke každému obrázku jsou známy informace o velikosti kódu (4 kategorie velikosti kódu v modulech), procentuální ploše QR kódu v obrázku, rovnoměrnosti osvětlení, zda je kód otočený, zda byla fotografie pořízena z pohledu kolmého na plochu QR kódu, nebo zda je fotografie rozmazaná. 430 obrázků je výběr z větší datové sady vyloučením obrázků, kde je plocha QR kódu menší než 30%, rozmazaných obrázků a obrázků, které mají nerovnoměrné osvětlení. Tento výběr obrázků je shodná datová sada se sadou užitou pro účely testování v [10]. Rozlišení obrázků je 600 x 337px. Ke každému obrázku je také přiřazena vzorová bitmapa QR kódu, který obsahuje.

Pro objektivní porovnání různých algoritmů pro detekci maticových kódů momentálně neexistuje žádná globálně uznávaná datová sada, ani metrika. V článku [10] jsou pro porovnávání zavedeny jisté metriky pro porovnání algoritmů. Hodnota metriky MX je procentuální podíl obrázků, které byly rozpoznány správně alespoň z X procent. Ve zmíněném článku jsou používány metriky M100, M99 a M95. Pro možnost srovnání tedy použiji stejné.

K určení správně rozpoznané části QR-kódu je v článku i mnou používán template matching. Tzn. vezmeme bitmapu získanou jako výstup z programu a vezmeme bitmapu vzorového kódu. Výstup z programu překryjeme vzorovou bitmapou a zjistíme počet shodných pixelů. Zapamatujeme si procento shodných pixelů. Poté vzorovou bitmapu posouváme a porovnáváme v každé možné pozici nad výstupem z programu. Totéž provedeme pro rotace a zrcadlení bitmapy. Celkem je 8 možností porovnání na každé pozici bitmapy výstupu z programu. Při porovnání hledáme nejlepší shodu.

Vzhledem k tomu, že procentuální hodnoty metrik a průměrný čas zpracování jednoho obrazu v milisekundách jsou podobná čísla, dovoluji si vkládat tyto dva typy veličin do společného grafu.

6.1 Testy parametrů algoritmu

Algoritmus zahrnuje velké množství parametrů, které bylo potřeba experimentálně vyhodnotit. K vyhodnocení výsledků sloužily výše popsané metriky a jelikož je práce zaměřena

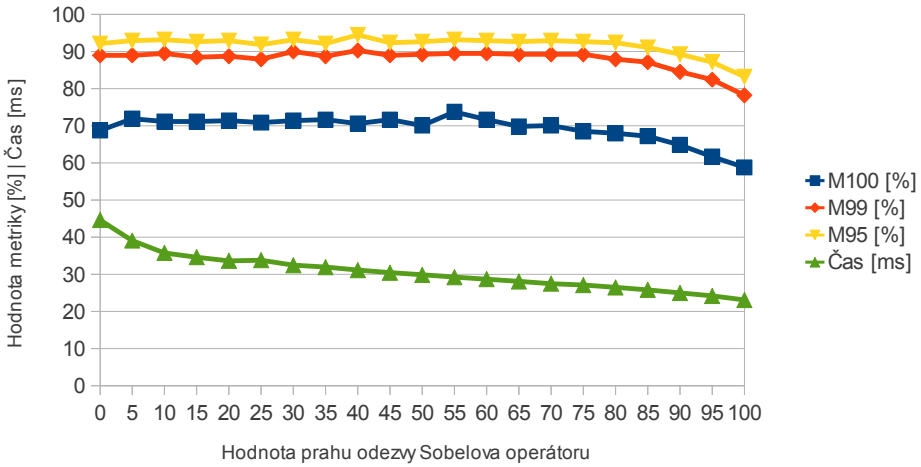
na zpracování v reálném čase, byl brán v potaz i průměrný čas zpracování jednoho obrazu. Ukázalo se, že nejlepší vypovídací hodnotu má metrika M100.

Testování probíhalo tak, že bylo vytvořeno základní nastavení podávající rozumné výsledky. Poté byly otestovány všechny parametry. Z výsledků byl vybrán nejlepší, parametr k němu náležící byl upraven na novou hodnotu a celý test se opakoval. Na základě dostatečného počtu iterací došlo k ustálení výsledků, které jsou zde prezentovány.

Při hledání nejlepších výsledků byl zásadní důraz kladen na zpracování v reálném čase. Potřeba rozpoznávat s co nejvyšší přesností byla až na druhém místě.

6.1.1 Rozdělení hranových bodů do binů

Před vlastním rozdělením hranových bodů do binů dochází k první filtraci hranových bodů. Jelikož jde o první filtr v algoritmu, hodnota má zásadní vliv na rychlost zpracování obrázku. V grafu 6.1 vidíme, že do cca hodnoty prahu 70 neklesá přesnost rozpoznávání. Nad hodnotu 70 jsou už pravděpodobně zahazovány i hranové body pocházející z maticového kódu. Za optimální hodnotu jsem vybral 55.



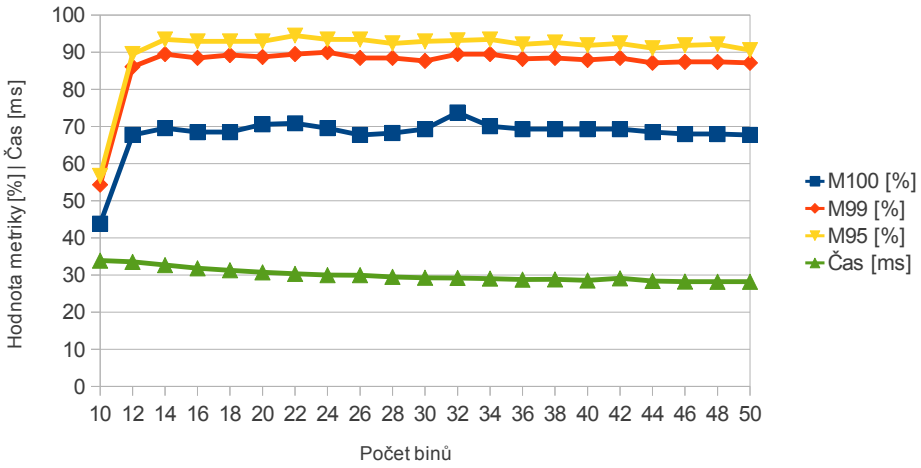
Obrázek 6.1: Graf závislosti metrik na prahu minimální odezvy Sobelova operátoru. Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent. Čas - průměrný čas zpracování jednoho obrázku.

Počet binů má vliv na omezení šířky oblasti, v rámci které jsou vykreslovány přímky v $\mathcal{TS}$ prostoru. Jsou přesněji určovány hraniční úhly jednotlivých sad rovnoběžek. To vede k mírnému zrychlení patrnému v grafu 6.2. Jako optimální hodnotu jsem vybral 32.

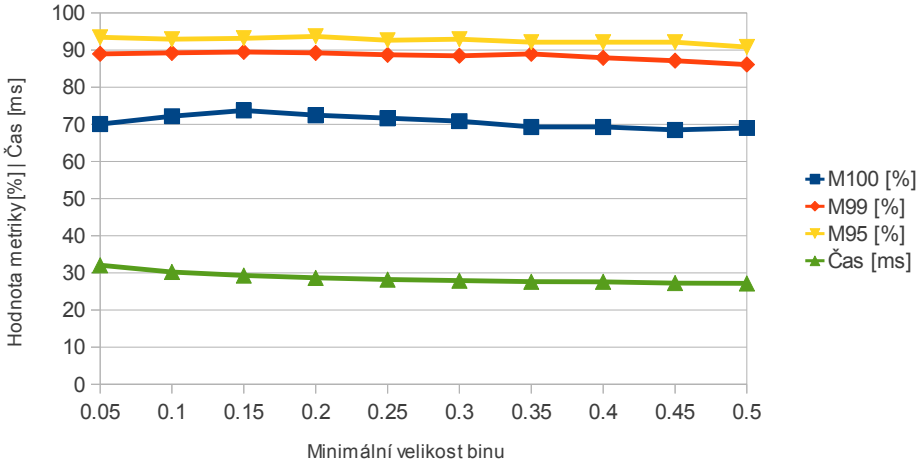
Další parametr určuje rozpětí hraničních úhlů rovnoběžek. Jedná se o minimální část velikosti maximálního binu. Více v sekci 4.1.3. Opět lze pozorovat mírné zrychlení při zvednutí hranice, ale také vidíme mírné zvýšení přesnosti rozpoznávání v okolí hodnoty 0.15.

6.1.2 Filtrace hranových bodů

Do algoritmu PClines vstupuje pouze omezený počet hranových bodů náležících jedné z sad rovnoběžek a s maximální velikostí odezvy Sobelova operátoru. V grafu 6.4 vidíme, že čas zpracování obrázku roste lineárně s počtem bodů, přesnost rozpoznávání však dosahuje maxima v blízkosti hodnoty 5000. Nižší hodnota znamená nedostatek informací k určení



Obrázek 6.2: Graf závislosti metrik na počtu binů. Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent. Čas - průměrný čas zpracování jednoho obrázku.



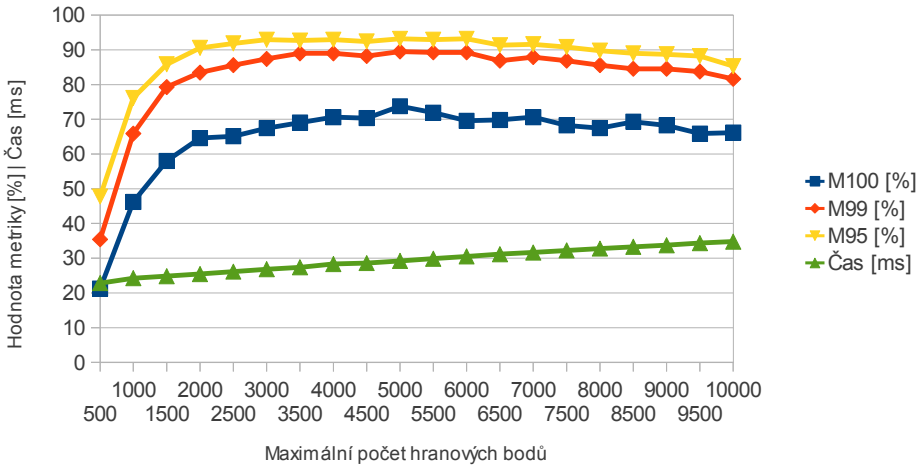
Obrázek 6.3: Graf závislosti metrik na minimální velikosti binu v okolí maxima. Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent. Čas - průměrný čas zpracování jednoho obrázku.

polohy rovnoběžek v maticovém kódu, protože jsou vybírány pouze nejvýraznější hrany, které se např. mohou vyskytovat pouze v malé části obrazu. Vyšší hodnota však zavádí do další části algoritmu hranové body, které nepatří maticovému kódu. Jedná se tedy o zanášení šumu do $\mathcal{TS}$ prostoru.

6.1.3 Rozlišení $\mathcal{TS}$ prostorů, parametr d

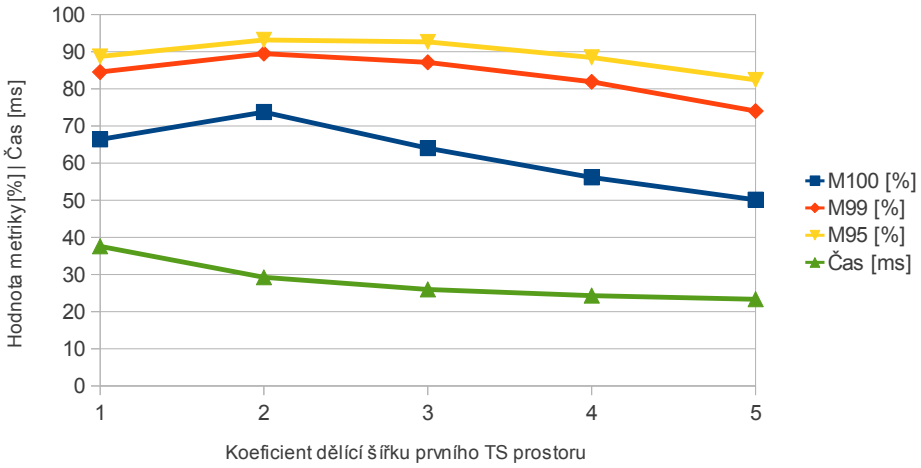
Základní rozlišení $\mathcal{TS}$ prostoru je maximum z výšky a šířky vstupního obrazu jako výška $\mathcal{TS}$ prostoru. Šířku definuje dvojnásobek parametru d (více zde 3.2.1). hodnota parametru d nelze vypočítat, proto je určována experimentálně. Jako základní hodnotu d jsem zvolil výšku $\mathcal{TS}$ prostoru.

Z principu fungování Houghovy transformace plyne, že šířka $\mathcal{TS}$ prostoru ovlivňuje



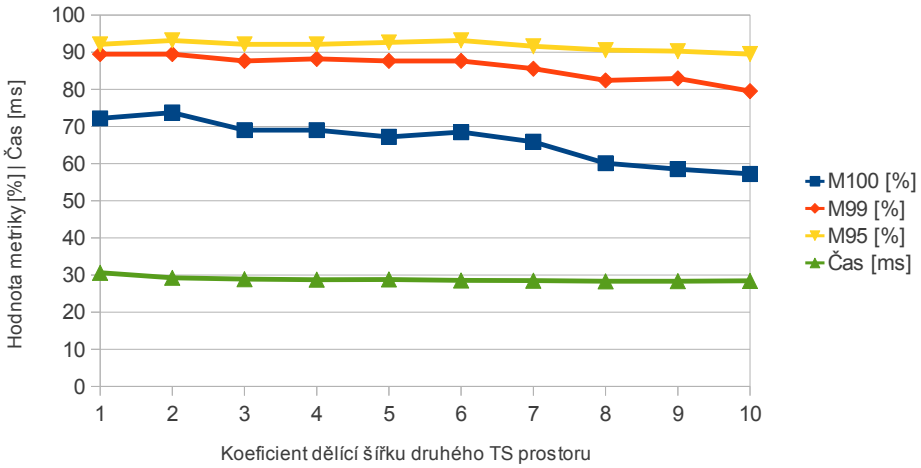
Obrázek 6.4: Graf závislosti metrik na počtu hran s největší odezvou. Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent. Čas - průměrný čas zpracování jednoho obrázku.

přesnost určení úhlu θ přímky při zpětné transformaci. A výška ovlivňuje hlavně přesnost určení vzdálenosti ρ od středu. Se změnou výšky $\mathcal{TS}$ prostoru nebylo experimentováno z důvodu jejího přesného definování v článku [8]. Hodnota parametru d byla dělena hodnotami uvedenými v grafu 6.5. Výsledky ukazují, že zmenšení parametru d na polovinu výšky $\mathcal{TS}$ prostoru přináší pozitivní výsledky jak v ohledu na přesnot rozpoznávání i na rychlost.



Obrázek 6.5: Graf závislosti na dělení parametru d v prvním $\mathcal{TS}$ prostoru. Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent. Čas - průměrný čas zpracování jednoho obrázku.

Podobné výsledky přináší graf 6.6 dělícího koeficientu v druhém $\mathcal{TS}$ prostoru. Z výsledků je ale patrné, že přínos v rychlosti je pouze nepatrný oproti prvnímu $\mathcal{TS}$ prostoru. Tento rozdíl plyne z počtu vkládaných bodů. V prvním $\mathcal{TS}$ prostoru jsou to tisíce, v druhém $\mathcal{TS}$ prostoru jsou to pouze desítky.



Obrázek 6.6: Graf závislosti na dělení parametru d v druhém $\mathcal{TS}$ prostoru. Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent. Čas - průměrný čas zpracování jednoho obrázku.

6.1.4 Lokální maxima v 1. $\mathcal{TS}$ prostoru

Počet hledaných lokálních maxim má vliv na přesnost nalezení přímky procházející v prvním $\mathcal{TS}$ prostoru skrz body reprezentující rovnoběžky. Je třeba si uvědomit, že k identifikaci takové přímky je vhodné použít nejlépe takový počet lokálních maxim, jaký je počet rovnoběžek. Při menším počtu lokálních maxim nemusí být určení polohy přímky dostatečně přesné. Při vyšším počtu lokálních maxim se mezi nimi vyskytují i maxima, která nereprezentují hledané rovnoběžné přímky. To může mít za následek nepřesné nalezení hledané přímky v $\mathcal{TS}$ prostoru.

Ze zmíněných důvodů správný počet lokálních maxim nelze dopředu přesně určit. Proto volím hodnotu na základě experimentálních výsledků nad testovací sadou. Vzhledem k tomu, že test probýhá nad QR kódy, lze očekávat výsledky mezi 22 a 178 vycházející z omezení velikosti QR kódu.

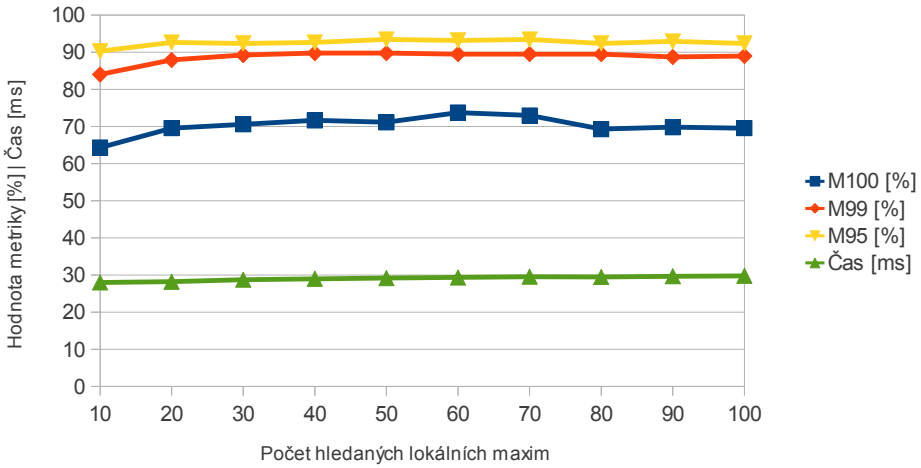
V grafu 6.7 vidíme, že pro nalezení hledané přímky stačí 20 lokálních maxim. Nejlepších výsledků algoritmus dosahuje až při hodnotě 60.

Při hledání lokálního maxima je důležitá velikost prohledávané lokality. Graf 6.8 ukazuje, že stačí malá velikost lokality, ideálně 5. Pro větší lokality dochází k tomu, že v prohledávané lokalitě se nachází více než jedno maximum(rovnoběžka). V takovém případě jsou v dané lokalitě zahazována menší lokální maxima.

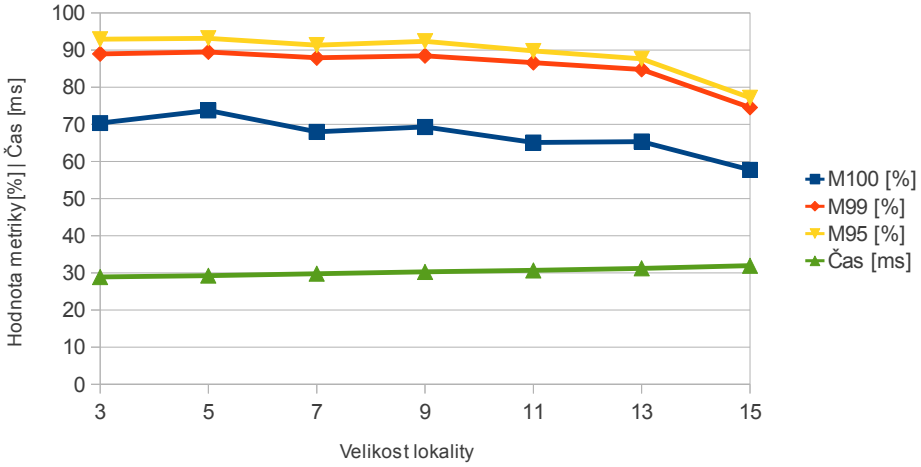
6.1.5 Detekce špiček

První parametr popisuje minimální velikost následující špičky jako část velikosti předchozí nalezené špičky. Hodnota musí být dostatečně malá, aby v profilech nebyly zahazovány špičky následující po velmi vysoké špičce. Naopak musí být hodnota dostatečně vysoká, aby tato část algoritmu odfiltrovala co nejvíce nízkých špiček, které většinou reprezentují šum. V grafu 6.8 lze pozorovat maximální hodnotu metriky M100 pro číslo 0,3.

Úkolem předchozí části algoritmu je hlavně detekce špiček. V následující části jsou nalezené špičky filtrovány. Filtr má za úkol odstanit špičky detekované v profilu v okolí maticového kódu. Tyto špičky se vyznačují tím, že jsou velmi nízké, protože jsou hrany



Obrázek 6.7: Graf závislosti metrik na počtu lokálních maxim hledaných v prvním $\mathcal{TS}$ prostoru. Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent. Čas - průměrný čas zpracování jednoho obrázku.

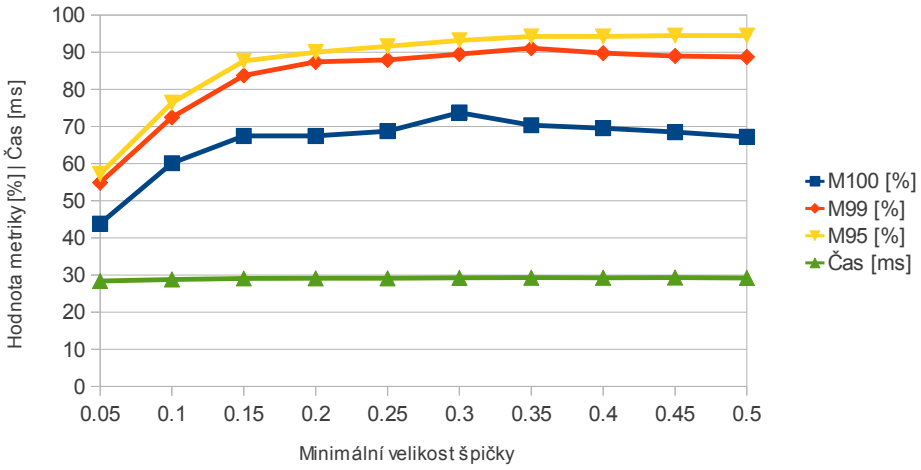


Obrázek 6.8: Graf závislosti metrik na velikosti lokality při hledání lokálních maxim. Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent. Čas - průměrný čas zpracování jednoho obrázku.

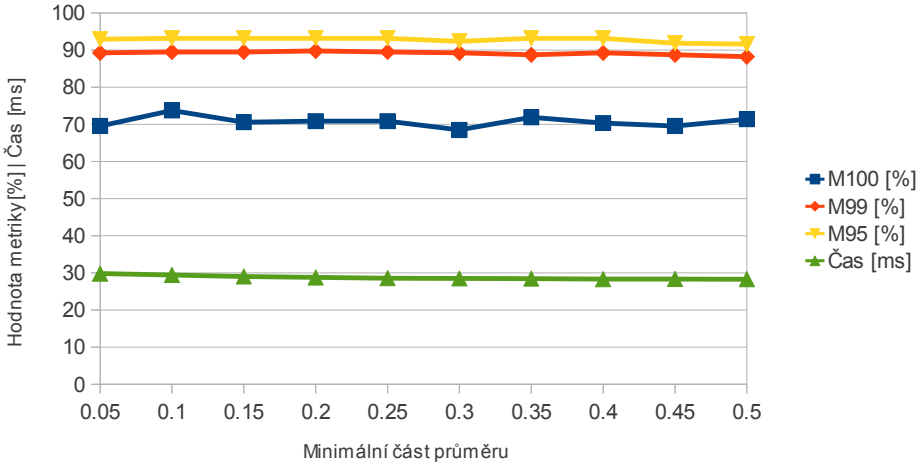
mimo maticový kód částečně odfiltrované. Hodnota v grafu 6.10 ukazuje poměrnou část z průměrné výšky špiček bez nejvyšší a nejnižší špičky. Graf také ukazuje, že přínos tohoto filtru není příliš vysoký, svědčí o tom i ideální hodnota zvolená 0,1.

6.1.6 Detekce inliers

Inliers jsou identifikovány na základě náhodně vybraných trojic z nalezených špiček. Počet zkoušených náhodných trojic má výrazý vliv na přesnost rozpoznávání. Vzhledem k tomu, že mezi detekovanými špičkami se stále nacházejí špičky, které nenáleží přímkám ze sady rovnoběžek, je třeba vyzkoušet značné množství náhodných trojic, než se nalezne dostatečně dlouhá posloupnost pro kterou platí cross-ratio rovnice. V grafu 6.11 vidíme, že pro vysoké počty trojic hodnota metrik M95 a M99 roste, avšak hodnota metriky M100 stagnuje od



Obrázek 6.9: Graf závislosti metrik na minimální části velikosti špičky. Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent. Čas - průměrný čas zpracování jednoho obrázku.



Obrázek 6.10: Graf závisloti metrik na parametru minimální části průměrné špičky. Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent. Čas - průměrný čas zpracování jednoho obrázku.

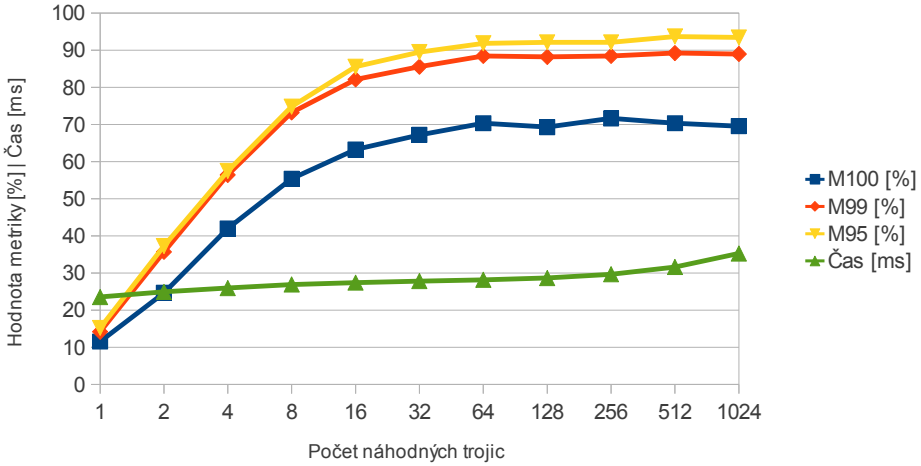
256 trojic. Za ideální hodnotu byl zvolen počet 200.

Graf 6.12 ukazuje hodnoty metrik pro různé velikosti maximální odchylky polohy špičky od polohy vypočítané. Nižší prahová hodnota než 0,7 zahazuje příliš mnoho špiček. Metrika M100 vykazuje v okolí hodnoty 1 mírně lepší výsledky, než hodnoty vyšší.

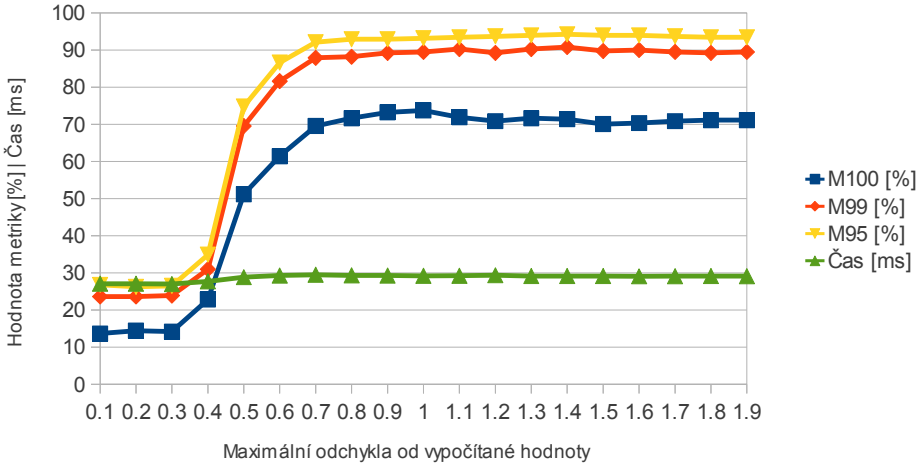
6.1.7 Parametry modelů

Protože není zaručeno, že každý model dospěje k dobrému výsledku, je vytvářeno více modelů. Výpočet modelu je časově náročný. V grafu 6.13 je vidět, že průměrná doba zpracování obrazu od dvaceti modelů roste velmi rychle. Hodnota metrik však již dále neroste.

V grafu 6.14 je vidět, že velikost počátečního kroku při tvorbě modelu není rozhodující. Toto je zapřičiněno tím, že je v jednotlivých iteracích krok dělen na polovinu. Metriky



Obrázek 6.11: Graf závislosti na počtu náhodně vybraných trojic. Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent. Čas - průměrný čas zpracování jednoho obrázku.

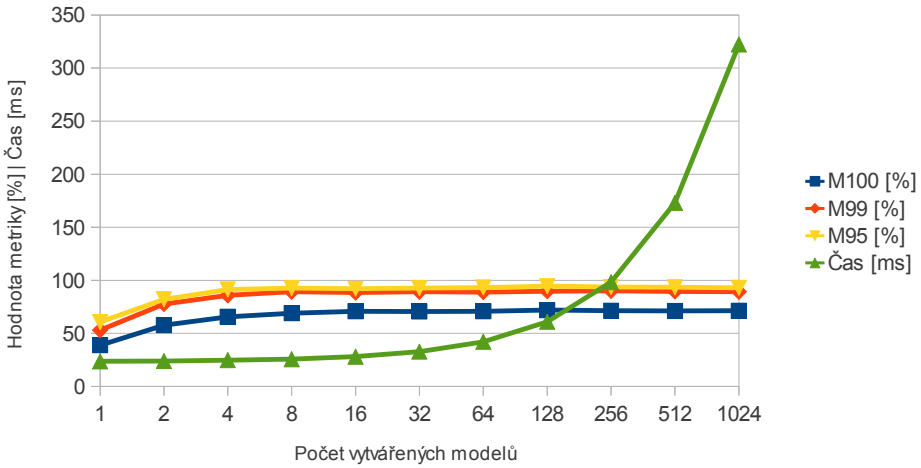


Obrázek 6.12: Graf závislosti maximální odchylky inliers. Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent. Čas - průměrný čas zpracování jednoho obrázku.

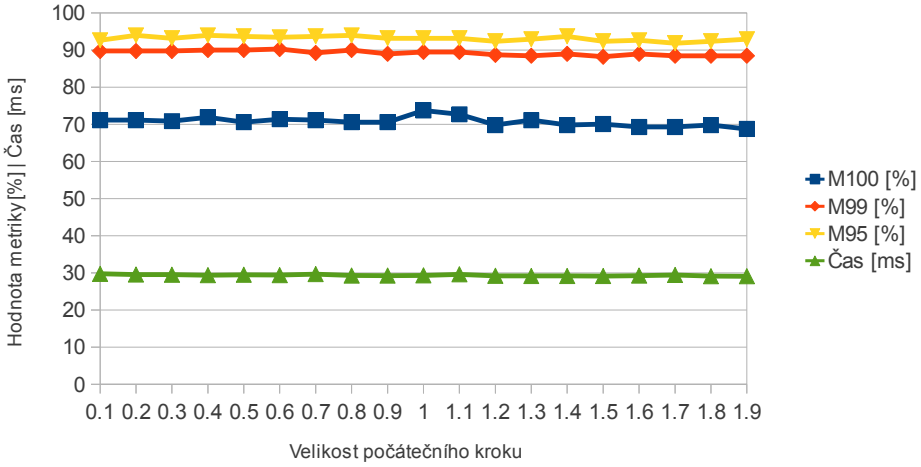
vykazují mírný nárůst hodnot v okolí velikosti kroku 1.

Graf 6.15 ukazuje, že počet iterací a tedy i dělení kroku, má výrazný vliv na čas výpočtu. Také je pozorovatelné, že krok velikosti 1 je příliš velký na přesné vytvoření modelu. Zvolil jsem počet iterací 4, protože vyšší počet iterací nevede ke zlepšení přesnosti rozpoznávání.

K určení počtu rovnoběžek generovaných modelem se vychází z počtu nalezených špiček. K tomuto počtu špiček je z každé strany dogenerováno při vytváření modelu dalších A + B špiček. Přičemž počet špiček A je zahrnut do výpočtu hodnoty metriky, ale špičky B zahrnuty do výpočtu nejsou. Předpoklad je takový, že špičky A rozšiřují sadu rovnoběžek na hranici maticového kódu (některé špičky se nepodařilo detekovat), proto je vhodné je zahrnout do výpočtu metriky modelu. Špičky B rozšiřují počet rovnoběžek dále pro extrémní případy, kdy bylo správně detekováno jen velmi málo špiček. Výsledky pro špičky A naleznete v grafu 6.16 a pro špičky B v grafu 6.17.



Obrázek 6.13: Graf závislosti metrik na počtu generovaných modelů. Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent. Čas - průměrný čas zpracování jednoho obrázku.

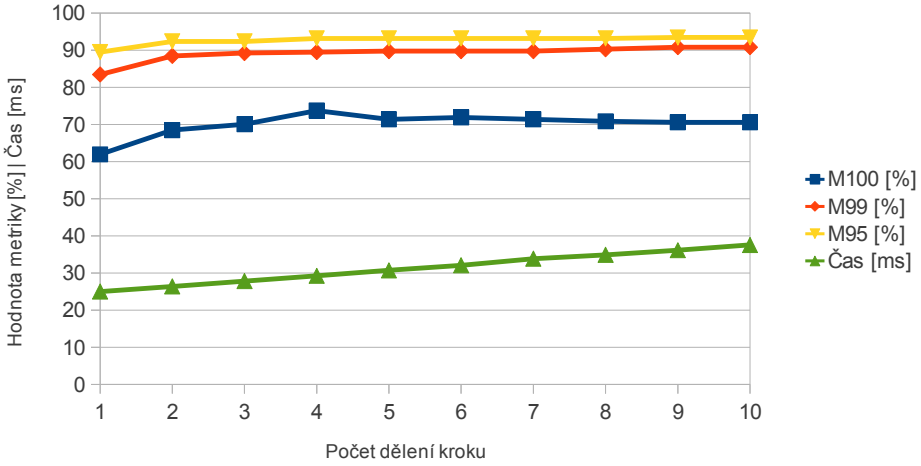


Obrázek 6.14: Graf závislosti metrik na velikosti počátečního kroku. Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent. Čas - průměrný čas zpracování jednoho obrázku.

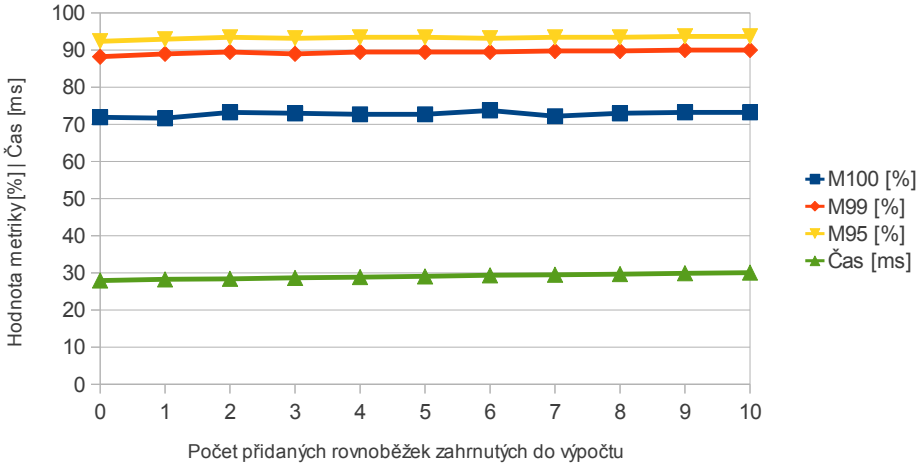
Z výsledků je patrné, že tyto parametry nemají velký vliv na výsledky algoritmu. Nejvhodnější hodnoty pro parametry A a B jsou 6 a 6.

6.1.8 Extrakce obrazu

Při extrakci obrazu je použito lokální prahování. Z grafu 6.18 je patrné, že lokální prahování má velký vliv na metriku M100. Rozdíl mezi metrikou M99 a M100 je zhruba 90%. Z tohoto plyne, že volba vhodného způsobu finální extrakce maticového kódu je velmi důležitá. Ukázalo se, že nejvhodnější velikost lokality je 21. Pozitivní zprávou z tohoto testu je, že přestože se hodnota 21 zdá vysoká, prakticky nemá vliv na průměrnou rychlost zpracování obrazu.



Obrázek 6.15: Graf závislosti metrik na počtu iterací (dělení délky kroku + 1). Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent. Čas - průměrný čas zpracování jednoho obrázku.

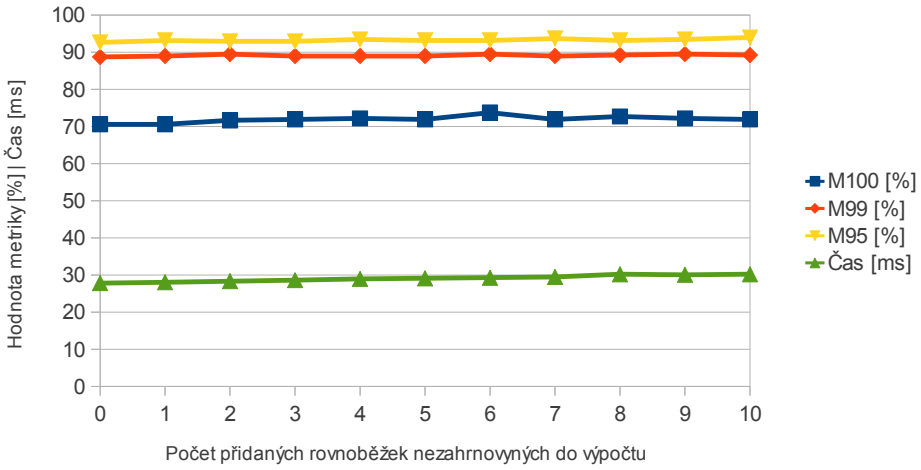


Obrázek 6.16: Graf závislosti metrik na počtu přidaných rovnoběžek, které jsou zahrnuty do výpočtu hodnoty metriky modelu. Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent. Čas - průměrný čas zpracování jednoho obrázku.

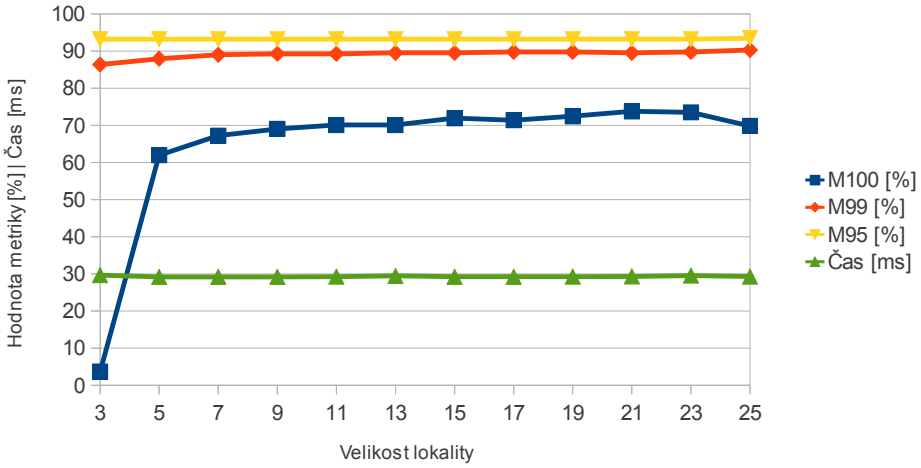
6.2 Test různých metrik pro hodnocení kvality modelu

Pro hodnocení kvality modelu bylo navrženo několik metrik a z nich vybrána metrika s nejlepšími výsledky.

Dle způsobu výpočtu lze navrhnuté metriky rozdělit do dvou kategorií. První typ metriky počítá chybu jako součet rozdílů poloh rovnoběžek vypočítaných modelem a rovnoběžek detekovaných v profilu jako špičky. Pro přehlednost pojmenuji tento typ metriky jako distanční. Druhý typ metriky bere v úvahu vlastnosti profilu. Jako pozitivní příspěvek do hodnoty metriky bere výšku profilu v místě vypočítaných rovnoběžek a jako záporný příspěvek do hodnoty metriky se bere hodnota profilu mezi vypočítanými polohami rovnoběžek. Tento typ metriky nazvu profilová metrika. Na základě popsaných typů metrik bylo vytvořeno a otestováno sedm následujících metrik.



Obrázek 6.17: Graf závisloti metrik na počtu přidaných rovnoběžek, které nejsou zahrnuty do výpočtu hodnoty metriky modelu. Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent. Čas - průměrný čas zpracování jednoho obrázku.



Obrázek 6.18: Graf závislosti metrik na velikosti lokality při lokálním prahování. Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent. Čas - průměrný čas zpracování jednoho obrázku.

1. distanční metrika - počítá vzdálenosti mezi inliers a nejbližší přímkou z modelu.
2. distanční metrika - počítá vzdálenost poloh mezi první inlier přímkou a přímkou v modelu vygenerovanou se stejným indexem, dále pokračuje přes všechny inliers tak, že jak u modelu, tak u inliers inkrementuje index o 1.
3. distanční metrika - počítá vzdálenosti mezi všemi přímkami vygenerovanými modelem a nejbližší detekovanou špičkou.
4. distanční metrika - počítá vzdálenosti mezi některými přímkami vygenerovanými modelem a nejbližší detekovanou špičkou. Přímky modelu jsou vybrány pouze ty nejbližší ke každé detekované špičce.

5. distanční metrika - je ekvivalentem metriky č. 2. Počítá vzdálenost poloh mezi přímkou v modelu a první inlier přímkou. Dále prochází všechny přímky modelu a počítá vzdálenost k ekvivalentní přímce v inliers.

6. profilová metrika - počítá hodnotu z profilu a poloh přímek z modelu takových, které mají indexy přímek odpovídající indexům přímek v inliers.

7. profilová metrika - počítá hodnotu z profilu v místech všech přímek generovaných modelem.

	Metrika kvality modelu						
Přesnost rozpoznání	1	2	3	4	5	6	7
M95	7.08%	81.88%	0%	51.18%	14.96%	88.45%	92.12%
M99	6.82%	70.34%	0%	46.98%	10.76%	78.74%	82.41%
M100	3.67%	49.86%	0%	35.17%	6.82%	57.48%	60.36%

Tabulka 6.1: Porovnání výsledků různých metrik.

V tabulce 6.1 vidíme, že metriky druhého typu, tedy ty, které berou v potaz profil mají větší úspěšnost. Poslední metrika podává výrazně lepší výsledky oproti prvním pěti metrikám. Proto byla zvolena jako metrika vhodná pro mnou navrhovaný algoritmus.

6.3 Vyhodnocení a porovnání výsledků

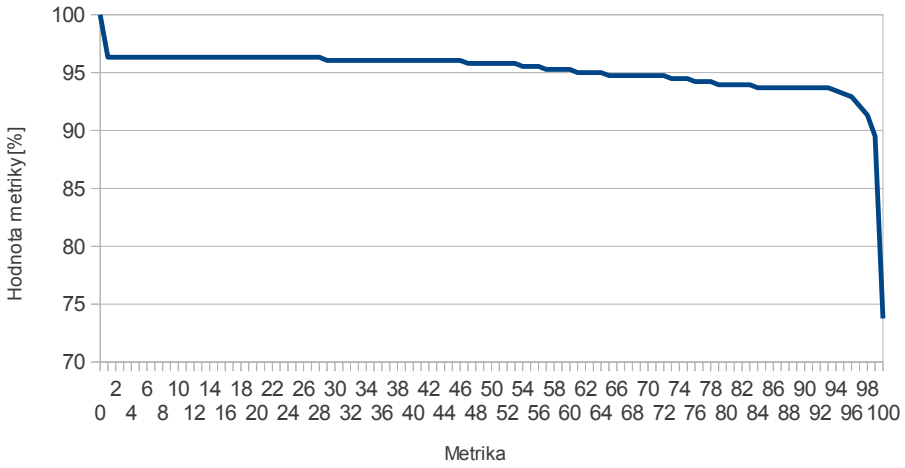
Výsledky algoritmu jsou porovnány s výsledky v článku [10] v tabulce 6.2. Vidíme, že algoritmus dosahuje lepších výsledků než nejpoužívanější opensource program ZXing. A to hlavně u obrázků s pootočeným maticovým kódem, nebo kódem vyfoceným z perspektivy. Algortimus dosahuje téměř shodných výsledků u obrázků bez rotace a perspektivy s variantou algoritmu M. Dubské, která také využívá PClines algoritmus. Ve srovnání s tímto algoritmem mnou navržený algoritmus dosahuje slušných výsleků u pootočených maticových kódů, ale u kódu focených s perspektivou zaostává.

Algoritmus	M. Dubská			ZXing			Mnou navržený		
Zkreslení obrazu	M100	M99	M95	M100	M99	M95	M100	M99	M95
bez rotace a persp.	88,5	97,3	98,2	88,5	97,3	97,3	85,8	97,3	98,2
rotace	68,4	97,3	98,9	53,7	58,9	62,1	74,3	86,4	93,6
perspektiva	89,1	95,5	95,5	69,1	74,5	74,5	65,4	84,5	93,6
rotace a persp.	78,6	96,4	97,3	9,8	9,8	9,8	67,8	88,0	91,6
celkem	81,6	96,0	97,4	55,3	60,2	60,9	73,7	89,5	93,1
celkem pro 1. variantu algoritmu popsanou v části 4.2.5							60,8	84,5	90,0

Tabulka 6.2: Porovnání dosažených výsledků s výsledky v článku [10].

V grafu 6.2 je také vidět porovnání variant mnou navržených algoritmů. Předposlední řádek ukazuje výsledky algoritmu založeného na vytváření modelů rovnoběžek popsaného v části 4.2.4. Poslední řádek ukazuje výsledky varianty založené na verifikaci detekovaných špiček popsané v části 4.2.5. Výsledky lze také interpretovat tak, že první varianta založená na individuálitě každého bodu má nižší přesnost než varianta druhá, která vytváří sadu rovnoběžek na základě vlastností celého profilu.

Výsledky algoritmu se dají také popsat pomocí křivky grafu. Jestliže je metrika MX definovaná jako procento obrazů, které bylo alespoň z X procent správně detekováno. Potom můžeme sestrojit graf funkce pro X = $< 0, 100 >$. Přičemž velikost kroku je libovolná, nemusí být ani celým číslem. Zvolil jsem velikost kroku 1 a výsledný graf je na obrázku 6.19.



Obrázek 6.19: Graf metrik MX pro X 0 až 100. Metrika MX - procentuální část obrázků, které byly správně rozpoznány alespoň z X procent.

S grafem 6.19 souvisí tabulka 6.3, která obsahuje vybrané zajímavé hodnoty. Lze pozorovat prudký pokles mezi metrikou 0 a 1. Ten je zapříčiněn odmítnutím algoritmu dále vyhodnocovat. Tzn. vstupní obrázek byl v průběhu provádění algoritmu vyhodnocen jako neobsahující maticový kód, nebo příliš nekvalitní. Další zajímavý bod je pokles nad metrikou M98, či M99. Jestliže na třech čtvrtinách ze vstupních obrazů jsou rozpoznány maticové kódy správně, pak zbývající cca čtvrtina vstupních obrázů měla nesprávně vyhodnoceno jen několik málo (ze tvaru grafu předpokládám nejčastěji jeden, nebo jednotky) modulů.

	Metrika						
	0	1	50	60	70	80	90
Hodnota metriky	100	96,3	95,8	95,2	94,7	93,9	93,7

	Metrika					
	95	96	97	98	99	100
Hodnota metriky	93,1	92,9	92,1	91,3	89,5	73,7

Tabulka 6.3: Výtah zajíavých hodnot z grafu 6.19.

6.4 Časová náročnost jednotlivých částí algoritmu

Vzhledem k tomu, že výsledná aplikace by měla pracovat v reálném čase, byl po celou dobu implementace a testování sledován průměrný čas zpracování jednoho obrazu. Při předpokladu, že chceme zpracovat 25 snímků za vteřinu máme 40 ms na jeden obrázek. Tabulka 6.4 ukazuje časovou náročnost jednotlivých částí algoritmu. V tabulce je také porovnání s podobným algoritmem M. Dubské popsaným v čnánku [10].

Část algoritmu	M. Dubská		Zde popsaný algoritmus	
	čas[ms]	čas[%]	čas[ms]	čas[%]
Extrakce a filtrace hran	6,7	46	7,8	28
Akumulace Houghova prostoru	3,1	21	8,4	31
Detekce lokálních maxim	0,8	6	2,3	8
Nalezení poloh modulů v obraze	3,7	25	8,1	29
Extrahovaní bitmapy	0,3	2	0,8	3
Celkem [ms]	14.6		27.5	

Tabulka 6.4: Tabulka porovnání průměrné doby trvání jednotlivých částí algoritmu.

Implementace M. Dubské byla testována na konfiguraci Intel Core 2 Duo E8300, 2x1GB 400 MHz RAM. Konfigurace mého testovacího počítače byla Intel Core 2 Duo E8400, 2x2GB 800 MHz RAM. Výsledky tedy nejsou přímo porovnatelné, ale poskytují náhled na časové náročnosti obou algoritmů.

Tabulka porovnává implementace při nastavení pro nejlepší možnou přesnost rozpoznání maticových kódů. Jednotlivé části algoritmu mají podoné rysy, avšak nejsou úplně shodné. Například část "Nalezení poloh modulů v obraze"je úplně odlišná. V mém algoritmu znamená využití nalezených lokálních maxim v druhém $\mathcal{TS}$ prostoru, řez a vytvoření profilu v prvním $\mathcal{TS}$ prostoru, vytvoření modelů a nalezení poloh modulů maticového kódu. V případě algoritmu M. Dubské se tato část nazývá "testování hypotézy"a její popis je v článku [10].

Z tabulky vidíme, že implementace M. Dubské je přibližně dvakrát rychlejší, avšak jednotlivé části zabírají procentuelně podobné množství času. Z tohoto usuzuji, že by bylo možné moji implementaci lépe optimalizovat. Nejnáročnější částí je dle očekávání akumulace Houghova prostoru, dále je to nalezení rovnoběžek v $\mathcal{TS}$ prostoru. Z této části trvá nejdéle nalezení špiček a vytváření modelů, která trvá 6,7 ms z 8,1 ms.

K celkovým časům v tabulkách je třeba připočítat načítání obrázků zprostředkované knihovnou OpenCV, které v průměru trvá 8,5 ms. Celková doba zpracování jednoho obrázku je tedy 36 ms, což odpovídá požadavku na zpracování v reálním čase.

Kapitola 7

Závěr

V rámci práce jsem se seznámil s různými typy maticových kódů a jejich zvláštnostmi. Podrobněji jsem se zaměřil na strukturu QR kódu, na jehož detekci a rozpoznání se částečně zaměřuje i navrhovaný algoritmus. Podrobně jsem pochopil princip Houghovy transformace, systém paralelních souřadnic, funkci rovnice cross-ratio a jejich možnosti využití pro detekci a rozpoznávání.

Byl navržen algoritmus pro detekci a rozpoznávání maticových kódů založený na algoritmu PClines. Algoritmus vytváří model rovnoběžných přímek procházejících hranami maticového kódu. Ukázalo se, že model vytvářený nad celou sadou rovnoběžek podává lepší výsledky, než algoritmus založný na detekci individuálních přímek. Algoritmus je odolný vůči zkreslení obrazu perspektivní projekci a otočení maticového kódu. Problémem zůstává kód na nerovném povrchu, rozmazaný obraz, nebo obraz výrazně zkreslený vlivem kulatosti čočky.

Algoritmus je implementován s ohledem na požadavek zpracování obrazu v reálném čase. Bylo experimentováno s nastavením jeho parametrů a výsledky byly diskutovány v kapitole 6. Dále byla vyhodnocena a porovnána jak přesnost ropoznávání algoritmu, tak jeho rychlost.

Výsledná implementace dosahuje uspokojivých výsledků ve směru přesnoti rozpoznávání. Na dané testovací sadě se podařilo 73,7 % maticových kódů rozpoznat bezchybně. Tyto výsledky by mělo být možné dále zlepšovat. Ve srovnání s algoritmem popsaným v článku [10], program dosahuje horších, avšak srovnatelných výsledků.

Program je schopen zpracovávat v reálném čase obrázky v nízkém rozlišení (600 x 337px, 27,5 ms + 8,5 ms načtení obrázku) i při horších osvětlovacích podmínkách, nebo při použití méně kvalitního fotoaparátu. Prostou úpravou parametrů algoritmu by bylo možné dosáhnout rychlejšího zpracování obrazu za cenu snížení přesnosti rozpoznávání.

Implementace zaostává v průměrné rychlosti zpracování obrazu. Zpracování jednoho obrazu trvá přibližně dvojnásobnou dobu oproti výsledkům popsaným v článku [10]. Z výsledků usuzuji, že by bylo možné moji implementaci lépe optimalizovat. K tomuto účelu by přispěl i lepší objektový návrh.

Pro zlepšení přesnosti rozpoznávání by bylo vhodné vyzkoušet vytváření modelů z více různých profilů ležících v okolí profilu vybraného pomocí užití druhého $\mathcal{TS}$ prostoru. Je ale otázkou, jak moc je informace přítomná v profilu reprezentativní a zda by nebylo vhodné vytvořit algoritmus využívající i informace přítomné v blízkém okolí profilu.

Literatura

[1] ISO/IEC 18004. Information technology - Automatic identification and data capture techniques - QR Code 2005 bar code symbology specification. 1. 9. 2006.

[2] Cross-ratio. In: Wikipedia: the free encyclopedia [online]. 14. 12. 2003, naposledy modifikováno 8. 11. 2011 [cit. 2011-26-12].
URL http://en.wikipedia.org/wiki/Cross-ratio

[3] ISO/IEC 24778. Information technology - Automatic identification and data capture techniques - Aztec Code bar code symbology specification. 15. 2. 2008.

[4] ISO/IEC 16022. Information technology - Automatic identification and data capture techniques - Data Matrix bar code symbology specification. 15. 9. 2006.

[5] Hough transform. In: Wikipedia: the free encyclopedia [online]. 18. 1. 2004, naposledy modifikováno 16. 12. 2011 [cit. 2011-26-12].
URL http://en.wikipedia.org/wiki/Hough_transform

[6] David Allais. In: Wikipedia: the free encyclopedia [online]. 2. 7. 2009, naposledy modifikováno 25. 5. 2011 [cit. 2011-26-12].
URL http://en.wikipedia.org/wiki/David_Allais

[7] Barcode. In: Wikipedia: the free encyclopedia [online]. 5. 6. 2002, naposledy modifikováno 12. 8. 2011 [cit. 2011-26-12].
URL http://en.wikipedia.org/wiki/Barcode

[8] Dubská, M.; Herout, A.; Havel, J.: PClines - Line Detection Using Parallel Coordinates. In *CVPR 2011*, Colorado Springs, US: IEEE CS, 2011, s. str. 1489–1494.

[9] Duda, R. O.; Hart, P. E.: Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM*, ročník 15, January 1972: s. 11–15, ISSN 0001-0782.
URL http://doi.acm.org/10.1145/361237.361242

[10] Herout, A.; Dubská, M.; Havel, J.: Real-Time Precise Detection of Regular Grids and Matrix Codes. In *v recenzním řízení*, CVPR 2012, 2012.

[11] Ritter, G. X. and Wilson, J. N.: *Handbook of computer vision algorithms in image algebra*. CRC Press, 2001, iSBN: 978-0-8493-0075-2.

[12] Technology, S.: Grid Matrix Code (GM) - Technology Character [online]. 2001-2010 [cit. 2011-26-12].
URL
http://www.syscantech.com/en/SyscanCode/Syscan_GM_TechCharacter.asp

Seznam příloh

A - Obsah CD
B - Manuál
C - Konfigurační soubor

Příloha A

Obsah CD

Bin - obsahuje spustitelný soubor přeložený na školním serveru Merlin. Soubor byl přeložen pomocí podmíněného překladu jako ladící, ale je vhodný i jako demonstrace funkce programu. Program je spustitelný příkazem `./qrcode`. Popis obsahu jednotlivých oken naleznete v části B.

Data - obrázky používané při vývoji algoritmu a programu.

- *QRcode_data* - obsahuje použitou testovací sadu vytvořenou M. Dubskou.
- *Pictures* - ladící a demonstrační obrázky.

Doc - adresář obsahuje programovou dokumentaci vygenerovanou programem Doxygen.

Src - obsahuje zdrojové soubory. Popis překladu a použití je popsán v části B.

Other - ostatní soubory:

- *tex* - obsahuje zdrojové soubory pro vytvoření technické zprávy a vygenerovanou technickou zprávu *projekt.pdf*.
- *plakat.pdf* - obsahuje vytvořený plakát.

Příloha B

Manuál

V adresáři `Src/nbproject/` nalezneme různé soubory Makefile. Program lze pomocí zvoleného makefile souboru podmíněným překladem přeložit ve dvou modech:

1. Test - slouží pro testovací účely.
   Překlad příkazem: `make -f nbproject/Makefile-Test.mk`
   Přeložený soubor poté naleznete v `Src/dist/Test/GNU-Linux-x86/qrcode`.

2. Debug - slouží pro účely ladění chyb, ale i jako demonstrace.
   Překlad pomocí: `make -f nbproject/Makefile-Debug.mk`
   Přeložený soubor poté naleznete v `Src/dist/Debug/GNU-Linux-x86/qrcode`.

Programy se spouštějí pomocí příkazu `./qrcode` bez parametrů. Všechny parametry jsou nastavitelné v souboru `Src/settings.h`, který je popsaný v části C.

Testovací program při spuštění očekává ve svém adresáři adresář `/Data/QRcode_data`. Pokud není `Src/settings.h` nakonfigurován jinak.

Program pro ladění očekává ve svém adresáři adresář `/Data/Pictures`. Výběr zobrazovaných obrázku je možné provést v části souboru `main.cpp`, která je překládána podmíněně s parametrem `DEBUG`.

V ladícím režimu se po spuštění objeví několik oken. Po stisknutí libovolné klávesy program zobrazí výstup z dalšího obrázku v pořadí. Popis oken je uveden níže:

1. Sobel - výstup ze Sobelova operátoru
2. Bins - velikosti binů, červená, zelená - vybrané maximální biny
3. Filter - výsledky filtrace pomocí binů
4. TS1-0, TS1-1 - první TS prostor a nalezená lokální maxima
5. TS2-0, TS2-1 - druhý TS prostor + nalezené maximum
6. TS0, TS1 - první TS prostor, přímka prochazejici body, reprezentující sadu rovnoběžek a vrcholy generované modelem
7. 1. TSLine, 2.TSLine - profil, žlutá - špičky, zelená - inliers, červená - trojice inicializující model, modře - modelem generované špičky
8. Grid - vstupní obraz, detekované rovnoběžky a vypočítané středy modulů

9. Extract1 - obraz složený z bodů přečtených v místech vypočítaných středů modulů
10. Extract - bitmapa získaná lokalním prahovaním
11. Extract - adjusted - bitmapa QR kódu nalezeného v předchozí bitmapě, ořezaného a otočeného pomocí třídy `QRExtractor`

Příloha C

Konfigurační soubor

Soubor `Src/settings.h` obsahuje veškeré nastavení programu. Parametry jsou stručně popsány v seznamu níže. Seznam je ve formátu "parametr - typická hodnota - význam".

SOBELKERNEL - 1 - velikost jádra Sobelova operátoru.

SOBELTHRESHOLD - 55 - minimální velikost odezvy Sobelova operátoru.

EDGESCOUNTMAX - 5000 - maximální počet hranových bodů vycházejících z filtru pro každou sadu rovnoběžek.

BINS - 32 - počet binů.

MINPOSMULT - 0.15 - prahová velikost binu (násobek maximálního binu).

NEIGHSIZE - 9 - velikost lokality v binech.

XSIZE - 640 - šířka obrázku s biny.

YSIZE - 480 - výška obrázku s biny.

D1DIVIDER - 2 - poměr výšky a šířky prvního Houghova prostoru.

D2DIVIDER - 2 - poměr výšky a šířky druhého Houghova prostoru.

LOCMAXCOUNT - 60 - počet hledaných lokálních maxim v prvním Houghově prostoru.

NEIGHSIZETS1 - 5 - velikost lokality při hledání lokálního maxima.

PARAMAXANGLE - M_PI_4 - maximální úhel přímky procházející body reprezentující rovnoběžky.

MINPEAKPART - 0.3 - hodnota pro výpočet prahové velikosti špičky (násobek předešlé).

MINPEAKAVERAGEPART - 0.1 - hodnota pro výpočet prahové velikosti špičky (násobek průměrné).

TRIPLETRYIES - 200 - počet náhodných trojic vybíraných z inliers.

CLEARANCECONST - 1 - maximální odchylka skutečné polohy špičky od vypočítané (výpočet iniliers).

MODELS - 20 - počet modelů.

STEPLEN - 1.0 - počáteční velikost kroku při tvoření modelu.

MOVES - 4 - počet iterací při tvoření modelu.

MODELADDITION - 6 - počet přidaných rovnoběžek zahrnutých do výpočtu metriky modelu.

MODELRESIZER - 6 - počet přidaných rovnoběžek nezahrnutých do výpočtu metriky modelu.

MINLINES - 5 - minimální počet nalezených špiček, jinak program končí.

QRMAXSIZE - 177 - maximální velikost QR kódu.

QRMINSIZE - 21 - minimální velikost QR kódu.

CORNERSQUARESIZE - 7 - velikost rohové značky QR kódu.

ERRTOLERANCY - 0.3 - maximální chyba v rohových značách, jinak program končí.

EXTRACTAREASIZE - 21 - velikost lokality při lokálním prahování.

EXTRACTSCALE - 6 - počet pixelů na modul výstupních obrázků QR kódu.

PATHDATASET - "cesta"- cesta vedoucí k testovaným obrázkům vyžaduje v adresáři jejich seznam a informace o nich. Vzor a popis naleznete v `Data/QRcode\_data`.

PATHDATAIN - "cesta"- cesta vedoucí ke vzorům maticových kódů.

DATAINEXTENSION - "jpg"- koncovka testovaných obrázků.

FAILMIN - 0.95 - minimální přesnost rozpoznání pro vybrané obrázky. Obrázky s hodnotami mezi FAILMIN a FAILMAX jsou uloženy do adresáře `/TestFailed/` (relativně od polohy programu).

FAILMAX - 0.98 - maximální přesnost rozpoznání pro vybrané obrázky. Obrázky s hodnotami mezi FAILMIN a FAILMAX jsou uloženy do adresáře `/TestFailed/` (relativně od polohy programu).

SHOWGRAPH - 0, nebo 1 - 1 pro zobrazení grafu metrik po testu.